MASTER EN SCIENCES INFORMATIQUES

La sécurisation des systèmes informatiques par la carte à micro-processeur

Serpe, Jean-Michel

Award date:
1988

Awarding institution:
Universite de Namur

FACULTÉS
UNIVERSITAIRES
N. D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

La sécurisation des systèmes informatiques par la carte à micro-processeur

Promoteur

Monsieur Jean Ramaekers

Mémoire présenté pour
l'obtention du grade de Licencié
et Maître en Informatique
par

Année académique 1987 - 1988

Jean-Michel Serpe

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Téiex 59.222 FAC NAM-B

Facultés Notre-Dame de la Paix Namur
Institut d'Informatique

rue Grandgagnage, 21, B-5000 NAMUR (Belgium)
☎ 081 / 22.90.65 Télex 59.222 FAC NAM-B

La sécurisation des systèmes informatiques par la carte à micro-processeur

Jean-Michel Serpe

Résumé

Un des problèmes des plus importants en informatique est la sécurité. En effet, toute autre découverte peut être inapplicable par suite d'une panne ou d'une attaque malveillante. Pour pallier en partie à ce problème, un outil est actuellement disponible : la carte à micro-processeur. Nous la présentons dans son utilisation avec serveur. Nous critiquerons et présenterons des améliorations et des perspectives probables.

Mots clés : authentification, carte à micro processeur, chiffrement, confidentialité, sécurité.

Abstract

One of the biggest problems in computer science is security. In fact any other discovery may become unusable due to a breakdown or malicious attacks. To resolve a part of this problem, a tool already exists : the smart card. We present it through its use in a network based on a server. We will criticize it and show many upgrades and different possible outlooks.

Key words : authentication, smart card, encryption, secrecy, security.

Mémoire présenté pour l'obtention du grade de Licencié et Maître en informatique.

Promoteur : J. Ramaekers

J'exprime toute ma gratitude à Monsieur J.Ramaekers, promoteur de ce mémoire, et à son assistante Madame C. Stas-Mahiat, pour l'attention qu'ils ont portée à la réalisation de ce mémoire et pour les excellents conseils qu'ils m'ont donnés.

Je remercie la direction de la société Bull CP8 pour m'avoir permis d'effectuer un stage très enrichissant. Mes plus vifs remerciements vont à Monsieur B. Rémy et à tous les membres de l'équipe logicielle pour l'aide précieuse, l'accueil chaleureux et leur sympathie que j'ai appréciés durant tout le stage.

Je tiens à exprimer mon entière reconnaissance à tous les professeurs de l'Institut d'Informatique pour l'enseignement donné ainsi qu'à toutes les personnes qui m'ont aidé et encouragé durant mes études et la réalisation de ce mémoire.

Namur, le 31 août 1988.

Jean-Michel Serpe

Table des matières

Introduction

Les problèmes de sécurité dans la gestion des applications informatiques prennent de nos jours un importance cruciale. Le danger est réellement présent à tel point que des revues, périodiques ou journaux "grand public", dont les publications ne portent pas sur l'informatique font de plus en plus état de faits plus ou moins inquiétants pour la survie de l'informatique, voire de notre société.

C'est ainsi que l'on peut lire quantité d'articles sur les virus, piratages où "intrusions" diverses dans des "super-computer" relativement sécurisés. On assiste aussi à des "enlèvements" de composants électroniques avec demande de rançon comme cela s'était passé en octobre 1986 près de Bruxelles.

Question de survie économique ? C'est très possible ; la dépendance des entreprises envers l'informatique croît de jour en jour. On estime à deux jours la durée maximale pendant laquelle une banque ou caisse d'épargne peut "survivre" sans ordinateur. Cette durée est portée à 3,3 jours pour les sociétés de distribution, 4,9 jours pour les socités de production et 5,6 jours pour les compagnies d'assurances. Que dire encore des pertes financiaires considérables dues au piratage de logiciel ?

Jean-Jacques Durré reprend une affirmation de Frans De Coster, directeur adjoint du Group 4 Securitas Alarm : "Un des éléments les plus importants en matière de sécurité informatique est le contrôle d'accès".

La carte à micro-processeur a été conçue spécialement dans le but de fournir un système d'authentification infraudable. Nous allons la présenter dans le cadre d'une utilisation dans un réseau. La carte offre d'autres fonctionalités que l'authentification. Elle dispose d'une mémoire qui permet de conserver des informations de manière sûre. Nous critiquerons cette solution et présenterons différentes améliorations.

Liste des abréviations utilisées :

EPROM : Eresable Programable Read Only Memory.

FIFO : First In First Out.

GOC : Générateur d'Octet Chiffrant.

MCS : Module de Contrôle et Sécurité.

NU : Non Utilisé.

PC : Personnal Computer

PROM : Programable Read Only Memory.

PSA : Processeur de Sécurité Généralisé.

OSA : Ordinateur de Sécurité Associé.

RAM : Random Access Memory.

ROM : Read Only Memory.

Chapitre 1

Présentation de la carte à micro-processeur

1.1. Introduction

La carte à micro-processeur a été développée dans le but d'être un support d'informations totalement sécurisé. Nous allons donc voir comment la carte gère elle-même la confidentialité des informations mémorisées dans sa mémoire. Pour plus de détails on consultera [M4 87].

1.2. L'aspect physique de la carte à micro-processeur

La carte à micro-processeur se présente sous la forme d'une carte de plastique de taille égale à celle des cartes de crédit. Cette carte possède en plus, dans son épaisseur, une puce qui comprend un micro-processeur et une mémoire. Ci-dessous nous vous présentons une carte à micro-processeur où l'on peut remarquer, dans le coin supérieur gauche, les contacts pour accéder à la puce ; dans le bas de la carte, un embossage (écriture en relief par déformation du plastique de la carte) donnant des informations sur le porteur de la carte et finalement un logo de l'émetteur de la carte (ici CP8).

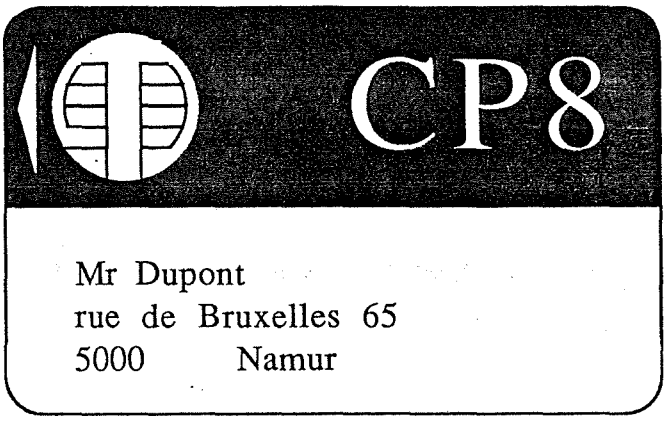


Figure 1.1. : La carte à micro-processeur

Il est à noter que la carte à micro-processeur peut aussi avoir une piste magnétique au verso,au même titre que les cartes de crédit classiques.

1.3. Le Principe général de la carte à micro-processeur

Le principe de fonctionnement de la carte à micro-processeur est schématisé comme à la figure 1.2. On peut remarquer que la connexion extérieure porte seulement sur le micro-processeur, il n'est donc pas possible d'accéder à la mémoire sans passer par le micro-processeur.

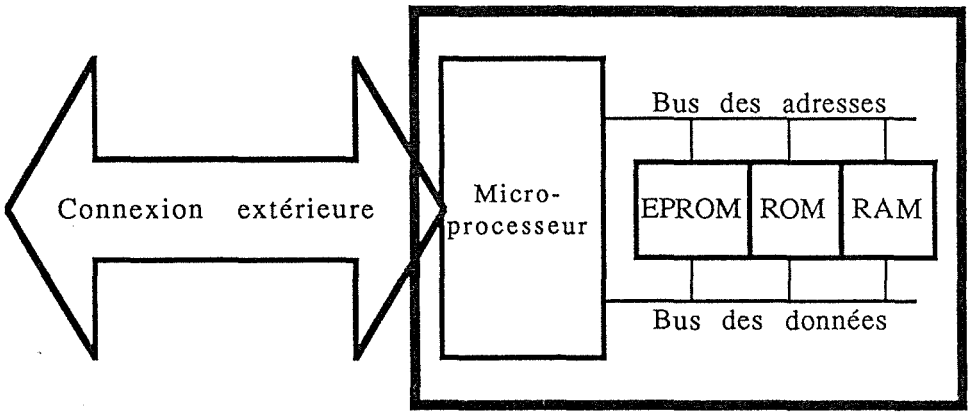


Figure 1.2. : architecture de la carte à micro-processeur.

Le micro-processeur est piloté par un programme mémorisé lors de la fabrication de la carte dans la partie ROM de la mémoire. Ce programme permet au micro-processeur de réagir aux commandes du monde extérieur tout en assurant la sécurité des données de la carte. Le programme détermine ce qu'on appelle le masque de la carte.

La partie EPROM de la mémoire contient les informations que l'application a mémorisées dans la carte ainsi que des informations nécessaires pour pouvoir assurer la sécurité de la carte. Il est à noter que cette mémoire, bien qu'elle soit de technologie EPROM, est utilisée en tant que PROM. Toute information inscrite dans cette mémoire y est définitivement.

La partie RAM est utilisée par le micro-processeur en tant que mémoire de travail.

1.4. Le cycle de vie d'une carte à micro-processeur

La carte à micro-processeur passe par plusieurs phases durant sa vie. Tout d'abord, il y a la fabrication de la puce. Dès la fabrication, des informations sont inscrites dans la puce pour assurer sa sécurité. Ensuite, il y a l'encartage de la puce. Cela consiste à poser la puce sur un support et à l'insérer dans l'épaisseur de la carte. Après cette opération vient la phase de personnalisation qui va inscrire les données nécessaires au fonctionnement de la carte dans le cadre d'une application. Vient ensuite la phase d'utilisation de la carte ou vie active et finalement la terminaison. Cette dernière peut survenir pour plusieurs raisons : la mémoire est saturée ou la carte est invalidée (tentative de fraude).

1.5. L'organisation de la mémoire EPROM

Les principes cités précédemment sont tout à fait généraux. En ce qui concerne l'utilisation de la mémoire EPROM, il a été décidé de la présenter telle qu'elle existe dans la carte masque 4.

Comme il l'a été dit précédemment, la mémoire EPROM fonctionne comme une mémoire PROM. A la fabrication, tous les bits de la mémoire sont à 1. Toute écriture consiste à "griller" certains bits. Un bit à zéro (grillé) ne peut plus revenir à 1.

Deux principes régissent l'accès à la mémoire EPROM. Tout d'abord, l'accès à la mémoire se fait en terme de mots de 32 bits ; ensuite, cette mémoire est divisée en zones qui se différencient par leur contenu et leurs droits d'accès. La protection des informations se fait à deux niveaux : règle de protection au niveau des zones et de chaque mot. Ainsi, selon les zones, un ou plusieurs des bits de poids fort de chaque mot sont réservés par le système pour assurer la sécurité du mot. Ces bits système sont nommés V, C et CA.

Le bit V permet de valider un mot. Dans ce cas, il n'est plus possible de modifier le contenu du mot.

Le bit C permet d'identifier le type de clé sous le contrôle de laquelle l'écriture a été faite (clé émetteur(1), clé porteur(0))[1]. Un émetteur est l'entité qui a distribué des cartes pour sécuriser l'accès à un service.

Le bit CA permet d'identifier quelle clé émetteur a été utilisée pour l'écriture, 2 émetteurs étant possibles pour une carte.

Nous allons maintenant passer en revue les différentes zones de la mémoire EPROM.

1.5.1. La zone secrète

La zone secrète est inaccessible au monde extérieur. On mémorise dans cette zone les clés et le jeu secret nécessaires pour assurer la sécurité de la carte. Trois types de clé sont utilisés : la clé de fabrication, la clé émetteur et la clé porteur. La clé de fabrication assure la sécurité de la puce pour la période s'étalant de la fin de la fabrication à la fin de la phase de personnalisation.

Les clés émetteurs peuvent être au nombre de deux (une par émetteur), nommées 1A et 1B. Elles servent aux prestataires[2] de services. La clé émetteur est nécessaire pour que le prestataire de services puisse inscrire des informations dans la carte.

La clé utilisateur permet l'identification du porteur de la carte par la carte elle-même. Cette clé peut être changée une fois par l'utilisateur.

Le jeu secret est nécessaire à la mise en oeuvre d'une fonction algorithmique. Cette fonction, cryptographique, est capitale dans la sécurité de la carte. Son utilisation sera développée ultérieurement.

Les bits système V et C sont présents dans chaque mot, ceci pour garantir que les secrets ne seront pas modifiés.

---

(1) Voir au paragraphe 1.5.1.

(2) On se basera dans la suite de l'exposé sur une configuration de type bancaire ; le banquier, prestataire de service et les clients porteurs de carte à micro-processeur.

1.5.2. La zone de fabrication

La zone de fabrication est de longueur fixe et localisée en fin de mémoire (9C0H). Cette zone contient différents pointeurs vers les différentes zones de la carte, un numéro de série ainsi que le numéro du fabricant. Finalement, plusieurs indicateurs permettent de spécifier l'accès de la zone de transaction(1) (LP et EP), l'étape de la vie de la carte (fabrication, personnalisation, activation de la deuxième clé porteur) et l'invalidation de la carte (certaines fonctions de la carte supprimées). L'accès est libre à cette zone en lecture. Toutes les informations, sauf le bit d'invalidation et le bit d'activation de la deuxième clé porteur, ont été écrites lors de la phase de personnalisation. Le bit système V est présent pour chaque mot.

1.5.3. La zone confidentielle

La zone confidentielle comprend des données qui ont été mémorisées lors de la personnalisation. Ces données ne sont accessibles qu'en lecture et sous condition d'avoir présenté une clé émetteur ou utilisateur. Les bits système V et C sont présents dans chaque mot.

1.5.4. La zone d'accès ou zone d'états

La zone d'accès est accessible seulement en lecture au monde extérieur, sous réserve d'avoir présenté une clé, et en lecture/écriture au micro-processeur, ceci pour la gestion et le contrôle des accès à la carte. Cette zone mémorise tous les accès faits à la carte sous contrôle d'une clé. Ceci permet de bloquer le fonctionnement de la carte en cas de trois erreurs consécutives sur présentation de clé utilisateur et une erreur sur clé émetteur. Il n'y a pas de bit système dans les mots de cette zone.

(1) Voir au paragraphe 1.5.5.

1.5.5. La zone transaction

La zone transaction est l'espace mémoire utilisé selon les besoins du ou des services auxquels la carte a été destinée. Les données qui y sont inscrites concernent l'application.

Les conditions d'accès à la zone transaction dépendent des deux indicateurs EP et LP mémorisés en zone de fabrication. Ces indicateurs imposent ou non la présentation d'une clé pour la lecture et l'écriture. Tous les mots ont les trois bits système V, C et CA.

1.5.6. La zone de lecture

La zone de lecture est destinée à l'enregistrement de données qui ne nécessitent pas de protection en lecture. Les bits système V et C sont présents dans chaque mot.

1.6. Le schéma général de la mémoire EPROM

Ci-dessous nous vous présentons un schéma général de la mémoire EPROM de la carte à micro-processeur.

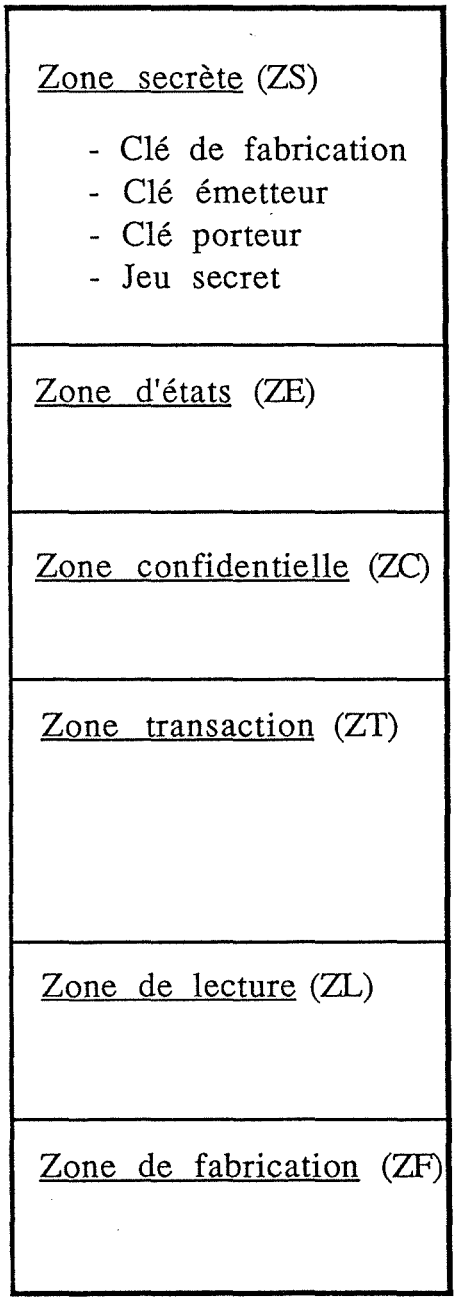


Figure 1.3. : Schéma général de la mémoire EPROM

1.7. Le tableau des règles d'accès aux différentes zones

Le tableau ci-dessous récapitule les règles d'accès aux différentes zones de la mémoire :

	Par le micro-processeur		Par le monde extérieur	
Zones	Ecriture	Lecture	Ecriture	Lecture
ZS	-	oui	oui (1)	-
ZE	oui	oui	-	oui/Clé
ZC	-	oui	-	oui/Clé
ZT	oui	oui	oui (1)	oui (1)
ZL	-	oui	-	oui
ZF	-	oui	oui (2)	oui

(1) Avec une présentation de clé selon les indicateurs de la zone de fabrication.

(2) Seulement pour le bit d'invalidation de la carte.

(3) Seulement pour le changement du code porteur

Figure 1.4. : Règles d'accès aux zones de la mémoire.

1.8. Les ordres exécutables par la carte

La carte à micro-processeur est conçue pour réagir à une petite dizaine d'ordres lancés à l'initiative du monde extérieur. Nous allons les passer en revue dans les paragraphes suivants.

1.8.1. La remise à zéro de la carte (RAZ)

L'ordre de RAZ permet d'initialiser le fonctionnement de la carte. La carte renvoie une série d'informations caractérisant son état.Il est à noter que cet ordre est hardware (exécuter une impulsion sur un des contacts de la puce).

1.8.2. La lecture

L'ordre de lecture permet de lire dans la mémoire EPROM jusqu'à 256 octets si les conditions d'accès à la zone demandée l'autorisent, sinon la carte devient muette : elle ne réagit plus tant qu'elle n'a pas reçu l'ordre d'initialisation.

1.8.3. L'écriture

L'ordre d'écriture permet d'écrire un mot (4 octets). Les conditions d'accès doivent être satisfaites, de même que le mot adressé doit être vierge, sinon la carte devient muette.

1.8.4. La présentation de clé

L'ordre de présentation de clé consiste à présenter une clé (clé émetteur ou clé porteur). C'est cet ordre qui assure une grande partie de la sécurité. En cas de clé émetteur fausse, de même qu'après trois présentations fausses de la clé porteur, la carte se bloque. Il est à noter qu'il est possible de débloquer la carte en lui présentant la clé porteur et la clé émetteur en même temps.

1.8.5. La validation d'accès en lecture

L'ordre de validation d'accès en lecture conditionne l'accès aux zones pour faire une lecture. Il y a donc trois ordres à mettre en oeuvre pour pouvoir faire une lecture :

- présentation de clé;
- validation d'accès;
- lecture.

1.8.6. La validation d'écriture

L'ordre de validation d'écriture consiste à positionner un bit système (le bit V) de telle manière que le mot écrit dans la carte ne peut plus être modifié.

1.8.7. La mise en oeuvre de la fonction algorithmique

L'ordre de mise en oeuvre est essentiel pour la sécurité d'une application de cartes à micro-processeur. La fonction algorithmique (ou cryptographique) est appelée "télépass". L'exécution de cette fonction s'effectue à partir de quatre éléments :

- le jeu secret mémorisé dans la zone secrète de la carte;
- un nombre aléatoire;
- l'adresse d'un mot en zone confidentielle ou zone de travail en zone de lecture;
- le contenu du mot.

Il est à noter que seuls le nombre aléatoire et l'adresse du mot sont communiqués comme paramètre à la fonction ; en effet, le jeu secret est mémorisé dans la zone secrète de la carte et accédé directement par le micro-processeur, de même que le contenu du mot spécifié.

1.8.8. La notion de carte mère et de diversification

Dans le cadre d'une application il y a bien entendu plusieurs cartes. L'émetteur des cartes (prestataire du service) doit pouvoir connaître la clé émetteur et le jeu secret de chaque carte. Deux solutions sont donc possibles. La

première serait d'avoir la même clé émetteur et le même jeu secret dans toutes les cartes. La seconde, serait de trouver un système de diversification d'une clé de base. La première solution a été rejetée pour des raisons de sécurité. En effet, si une personne parvenait à connaître le secret d'une carte, elle aurait par conséquent le secret de toutes les cartes. De ce fait, on a adopté la seconde solution. C'est ici que la notion de carte mère prend place.

La carte mère est une carte comme les autres. Comme on peut le voir sur la figure 1.5. elle contient un jeu secret appelé jeu secret de base ou jeu secret mère. Le jeu secret de la carte fille est le résultat de la fonction télépass exécutée avec, comme nombre aléatoire, son numéro de série et une adresse de la carte mère contenant un "Pattern" en fonction du type de secret. Ce principe est appliqué à toutes les clés de la carte carte fille.

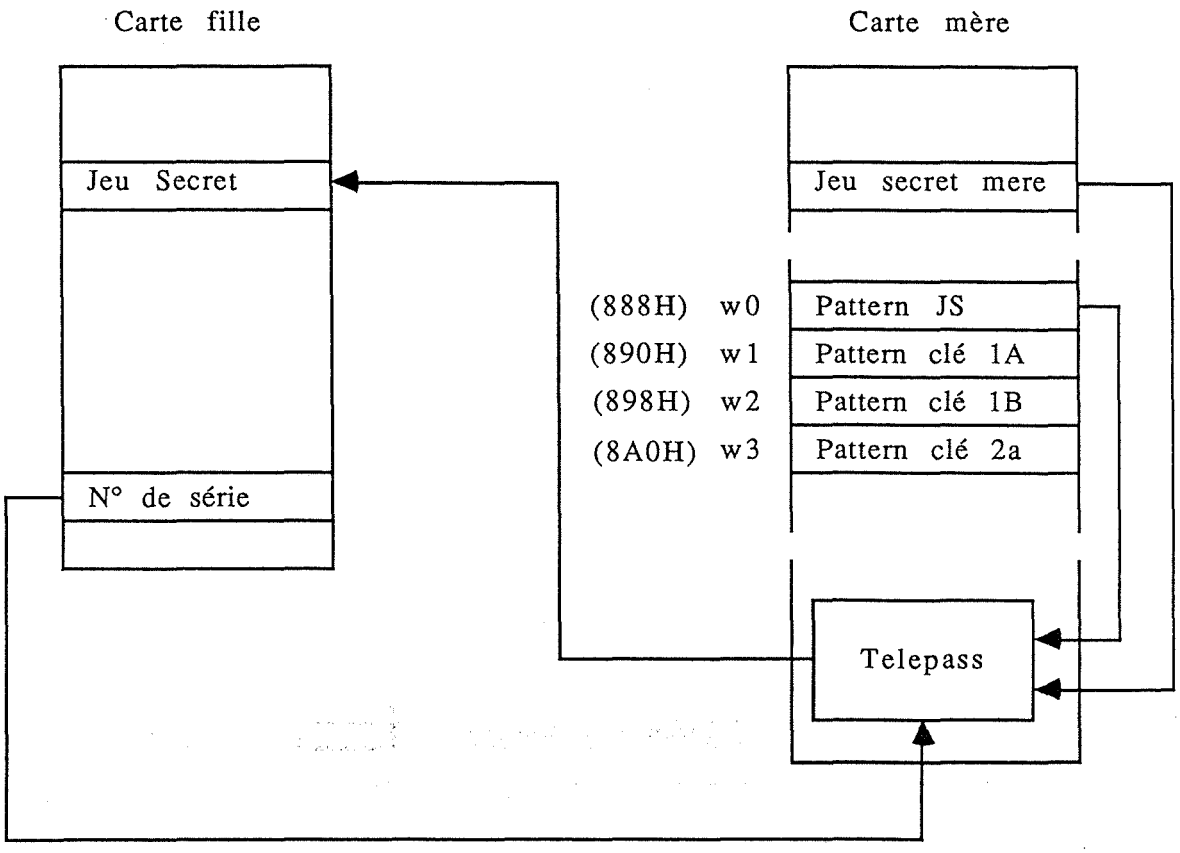


Figure 1.5. : principe de diversification.

Chapitre 2

Le module de contrôle et de sécurité

2.1. Introduction

Le module de contrôle et de sécurité ou MCS est un composant électronique à 28 broches. Son objectif est d'offrir un support totalement sécurisé pour les secrets de base d'une application (secret mère) ainsi que la protection de la fonction télépass. Pour plus de détails on consultera [MCS 87].

Il faut se rappeler que la carte est protégée par plusieurs clés dont la clé émetteur. Il est donc indispensable, pour un prestataire, d'en disposer pour pouvoir accéder à la mémoire de la carte. C'est dans ce but que le MCS a été conçu.

Le MCS permet la réalisation d'autres fonctions très importantes au niveau de la sécurité. Ces fonctions sont l'authentification de la carte, la certification du contenu d'un mot, le calcul d'une clé de chiffrement, la mise à disposition de la clé émetteur à l'ordinateur qui doit traiter les cartes. Ci-dessous, nous vous présentons l'aspect physique du MCS.

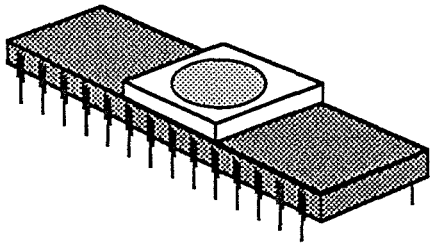


Figure 2.1 : Aspect physique du MCS.

2.2. Le principe du module de contrôle et de sécurité

Le MCS comprend, tout comme la carte à micro-processeur, un micro-processeur et une mémoire. Il possède bien entendu un programme interne qui lui permet de réaliser une série d'ordres. La mémoire EPROM n'est divisée qu'en deux zones : la zone d'utilisation et la zone de fabrication.

2.2.1. Structure du mot

Le MCS comprend 62 mots dans sa zone d'utilisation. Chaque mot est divisé en deux zones : une zone de 3 bits système et une autre de 29 bits d'information.

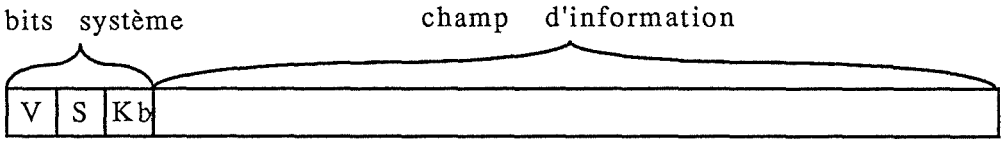


Figure 2.2 : Structure du mot.

Le bit système V signale si le mot est validé ou non. Si le mot n'est pas validé (V=1), les autres bits système n'ont pas de signification et le mot peut toujours être accédé en lecture ou écriture. Si le mot est validé (V=0), les deux autres bits système prennent leur signification et le contenu du mot ne peut plus être modifié.

Le bit système S permet de dire si le mot peut être lu (S=1) ou non (S=0). Dans ce dernier cas, seul le micro-processeur peut accéder au mot pour ses traitements propres.

Le bit système Kb, qui n'a sa signification que si le bit système S est égal à 1 (sinon il fait partie des bits d'information). Il spécifie que le mot débute un bloc secret (Kb=0) ou est un mot courant du bloc secret (Kb=1).

2.2.2. Le bloc secret

Le bloc secret est une structure d'information sur 6 mots permettant de mémoriser un jeu secret de base (ou jeu secret mère) et les informations nécessaires à sa diversification et sa sécurité. Ci-dessous, le schéma de la structure d'un bloc secret.

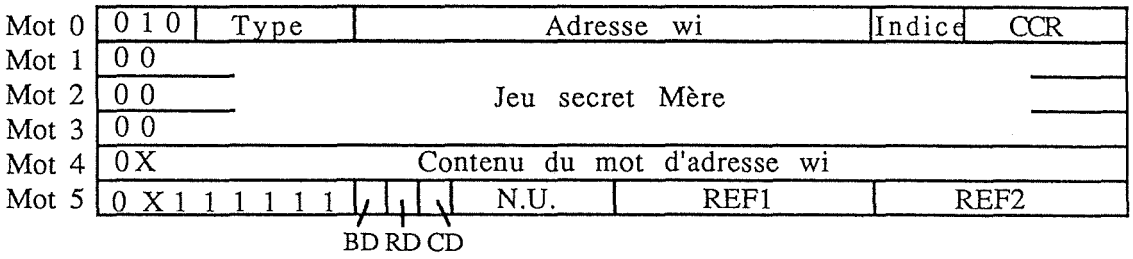


Figure 2.3. : Structure d'un bloc secret.

Nous allons maintenant passer en revue toutes les zones d'un bloc secret.

2.2.2.1. Le type et l'indice

Type et indice sont deux champs respectivement de 5 et 3 bits qui permettent d'identifier un bloc secret.

Le **type** fait correspondre le jeu secret mère à un parc de cartes (ensemble de cartes issues de la même carte mère).

L'**indice** donne le "type" de clé à laquelle est destiné le jeu secret :

Indice	Clé
000	Clé de chiffrement
001	Clé de certificat
010	Clé émetteur 1A
011	Clé émetteur 1B
100	Clé certificat sur 2 Octets

Figure 2.4. : Correspondance Indice / type de clé

2.2.2.2. L'adresse Wi

L'adresse Wi est une zone qui contient l'adresse du mot de la carte mère qui a été utilisée pour la diversification du secret pour lequel le bloc est destiné. Cette zone n'est pas toujours utilisée.

2.2.2.3. La zone CCR

La zone CCR contient un code détecteur d'erreur sur 5 bits. C'est le résultat de la division de tous les bits sauf du bit V par le polynôme $G(x) = x^5 + x^2 + 1$

2.2.2.4. Le jeu secret mère

Le jeu secret mère est identique à celui qui a été mémorisé dans la carte mère utilisée en phase de personnalisation.

2.2.2.5. Le contenu du mot d'adresse Wi

Le contenu du mot d'adresse Wi correspond au mot d'adresse Wi de la carte mère qui a servi à la diversification. Cette zone n'est pas toujours utilisée. Dans ce cas le quatrième mot est égal à 7FFF FFFF.

2.2.2.6. L'indicateur BD

L'indicateur BD permet, lors de la commande de "sélection de bloc"(1), de provoquer ou non le calcul de la diversification du jeu secret (BD=1, pas de diversification, BD=0 : diversification). Si la diversification est demandée, elle se fait à partir des zones Wi et du contenu du mot d'adresse Wi.

2.2.2.7. L'indicateur RD

L'indicateur RD permet d'autoriser ou non la lecture du résultat d'un calcul. Si RD est égal à 1, les résultats sont lisibles, sinon on ne peut faire qu'une comparaison d'une valeur avec le résultat.

2.2.2.8. L'indicateur CD et les zones Ref1 et Ref2

L'indicateur CD permet de limiter l'usage de la fonction algorithmique. Si CD=1, alors l'octet de poids fort du mot sur lequel l'algorithme est effectué, noté OF, doit être tel que (REF1 AND OF) = REF2.

Si l'indicateur CD=0, alors l'adresse du mot sur lequel l'algorithme est effectué doit être égale à REF1_REF2.

2.3. L'utilisation des indicateurs BD, RD et CD

La combinaison des indicateurs BD, RD et CD est essentielle. Elle permet au responsable de la sécurité d'une application de donner seulement certains droits à un centre d'exploitation. En effet, on peut par exemple concevoir un bloc pour réaliser l'authentification de la carte et cela de telle manière qu'il ne soit pas possible de faire autre chose, tel que connaître une clé émetteur ou pouvoir générer une clé de chiffrement.

Une description d'une conception classique de blocs sera développée au point 2.5. ainsi que leur objectif. Maintenant nous allons vous présenter les ordres disponibles dans un MCS.

(1) Sera présenté au paragraphe 2.4.5.

2.4. Les ordres du MCS

2.4.1. L'ordre de remise à zéro (RAZ)

Cet ordre hardware, permet de réinitialiser le MCS. Ce dernier renvoit une suite d'octets caractérisant son état.

2.4.2. L'ordre de lecture

L'ordre de lecture permet de lire jusqu'à 256 octets. Les octets appartenant à un mot déclaré secret (bit S=0) sont forcés automatiquement à 0 binaire. Ceci est essentiel et permet d'assurer que les jeux secrets mémorisés dans les blocs ne sont jamais "visibles" par le monde extérieur.

2.4.3. L'ordre d'écriture

L'ordre d'écriture permet d'enregistrer un mot dans la mémoire d'un MCS, la seule restriction est qu'il n'est pas possible d'écrire un mot déjà validé. Le bit système V n'est pas positionné par un ordre d'écriture.

2.4.4. L'ordre de validation

L'ordre de validation a pour but de positionner le bit système V d'un mot et ainsi de protéger l'information qu'il contient.

2.4.5. L'ordre de sélection d'un bloc secret

L'ordre de sélection d'un bloc secret permet, comme son nom l'indique, de sélectionner un bloc secret, donc un jeu secret. Il est à noter que c'est à ce moment que le MCS effectue le calcul de diversification si l'indicateur BD l'impose (BD=0). La sélection se fait sur base du Type et de l'Indice.

2.4.6. L'ordre de calcul

L'ordre de calcul permet d'exécuter la même fonction algorithmique que celle de la carte à micro-processeur. Le jeu secret utilisé est celui qui a été sélectionné la dernière fois (éventuellement diversifié lors de sa sélection).

2.4.7. L'ordre de calcul chaîné

L'ordre de calcul chaîné a exactement la même fonctionnalité que l'ordre de calcul à la différence près que le résultat est combiné avec le résultat précédent par une opération XOR. Pour cette raison, l'ordre de sélection suivi d'un ordre de calcul chaîné n'a pas de sens (résultat imprévisible). On doit donc toujours commencer une chaîne de calculs par un ordre de calcul (simple).

2.4.8. L'ordre de repli de résultat

L'ordre de repli de résultat permet de réduire la taille d'un résultat d'un ordre de calcul (64 bits) à une longueur L telle que $0 < L < 64$. A cette fin un algorithme simple est effectué. Il consiste, comme le présente la figure 2.5., à faire (64 - L) décalages et à chaque fois de réintroduire le bit "expulsé" par une opération XOR sur le $L^{ième}$ bit du résultat intermédiaire.

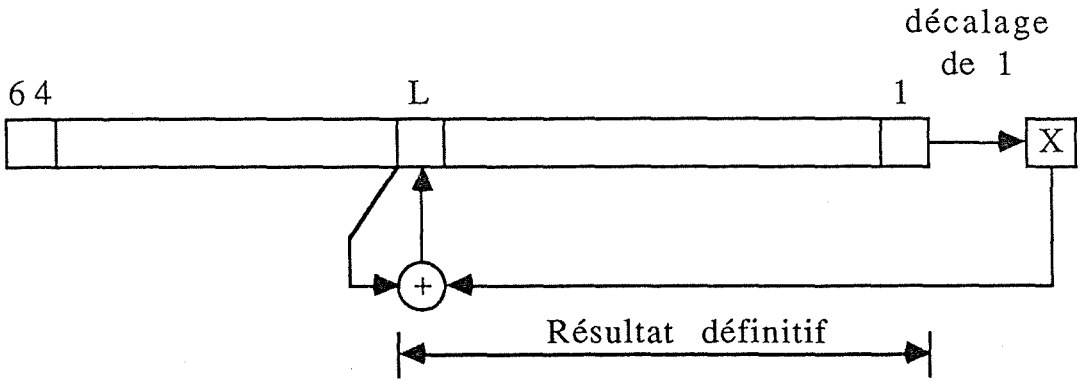


Figure 2.5. : Mécanisme du repli d'un résultat

2.4.9. L'ordre de lecture de résultat

L'ordre de lecture de résultat permet au monde extérieur de connaître le résultat d'un calcul à condition que l'indicateur du bloc qui avait été sélectionné pour le calcul l'autorise (RD=1).

2.4.10. L'ordre de comparaison de résultat

L'ordre de comparaison de résultat permet au monde extérieur de comparer une donnée en sa possession (par exemple le résultat d'un calcul de la carte) avec le résultat du calcul dernièrement effectué par le MCS.

2.5. La sécurité d'une application

En général, toute application comprend un prestataire de services et une multitude de porteurs d'une carte à micro-processeur qui utilisent le service. Il y a donc souvent plusieurs lecteurs de cartes reliés à un serveur(1). La sécurité de l'application repose notamment sur celle établie sur la "connexion" entre le lecteur et le serveur. Pour assurer cette fonction de sécurité, quatres principes sont utilisés; il s'agit de l'authentification, la certification du chiffrement et de la signature.

2.5.1. L'authentification

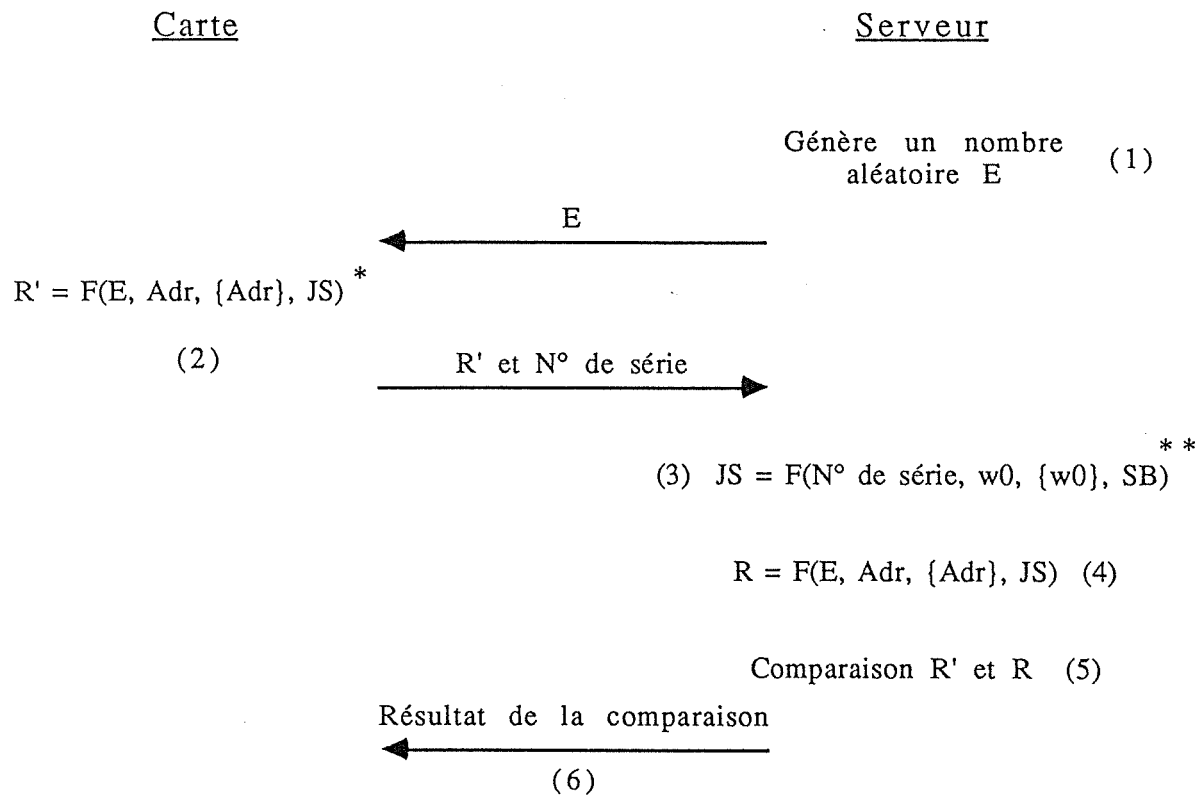
Plusieurs types d'authentification existent dans le cadre d'applications sur base de la carte à micro-processeur. C'est ainsi qu'il y a l'authentification du terminal, du porteur par rapport à la carte et de la carte par rapport au serveur. C'est cette dernière que nous allons développer.

L'authentification de la carte par rapport au serveur c'est, somme toute, la vérification de l'appartenance de la carte au parc de cartes de l'application. L'authentification est donc une première protection du serveur contre des personnes n'ayant pas de droit d'accès au service.

(1) Il est à noter que l'on peut avoir un réseau de serveurs. Ce point sera développé au chapitre 5.

L'authentification est un principe relativement facile à réaliser. Dans un premier temps, comme le décrit la figure 2.6., le serveur génère un nombre aléatoire qu'il communique à la carte (1) La carte exécute la fonction algorithmique, appliquée à un mot connu aussi du serveur. Le résultat ainsi que le numéro de série sont envoyés au serveur (2)

De son coté le MCS recalcule le jeu secret de la carte à partir du jeu secret mère et du numéro de série de la carte qui vient d'être communiqué (3). Le même calcul que celui de la carte est ensuite réalisé (4). Il reste à comparer les résultats (5). S'ils sont égaux, c'est que le jeu secret de la carte était le même que celui qui a été généré (propriété de la fonction algorithmique) et donc que la carte fait bien partie du parc de cartes de l'application. Le résultat est communiqué à la carte (plus précisément au terminal qui a reçu la carte).



* Adr et {Adr} sont connus du serveur, souvent un mot contenant les droits d'accès.

** Voir le principe de diversification au paragraphe 1.8.

Figure 2.6. : Authentification.

Il est à noter que la fonction algorithmique utilise un nombre aléatoire, ce qui garantit que la réponse de la carte est différente à chaque authentification. De ce fait, si la réponse est interceptée sur la ligne par un espion, elle ne lui est d'aucune utilité.

La description du bloc secret mémorisé dans le MCS pour réaliser cette fonction sera abordée dans le paragraphe suivant.

2.5.2. La certification

La certification a pour but de garantir au serveur la présence d'une information à une adresse précise de la mémoire EPROM de la carte. Son objectif est donc la protection du serveur contre un porteur de carte à micro-processeur qui a droit d'accès au service mais qui tente de tricher, par exemple, en empêchant l'écriture de la transaction dans la carte. Dans ce cas il lui est impossible de produire un certificat.

Le certificat est simplement le résultat de la fonction algorithmique de la carte appliquée à un nombre aléatoire, une adresse de la mémoire EPROM, son contenu, et, bien entendu, le jeu secret. Comme au niveau du serveur on peut connaître le jeu secret, il est possible de vérifier un certificat, le serveur connaissant aussi l'adresse du mot et son contenu théorique à prouver. Un certificat est souvent demandé après une lecture ou une écriture pour la garantir.

On peut remarquer que l'authentification est un cas particulier de la certification. La procédure est donc tout à fait similaire; seule l'adresse du mot (et son contenu) sur laquelle la fonction algorithmique est appliquée change.

Nous allons maintenant voir comment on peut établir un bloc dans le MCS qui permet de réaliser cette fonction. Lors de l'utilisation du bloc on distingue principalement trois étapes dans la réalisation d'un contrôle de certificat.

La première étape se charge du calcul du jeu secret de la carte fille, la seconde du calcul du certificat et la troisième de la comparaison. Le bloc secret doit donc provoquer le calcul de diversification du jeu secret mère lors de sa sélection du bloc, (indicateur BD=0) et par sécurité, ne pas permettre de lire le résultat d'un calcul (certificat). La comparaison est largement suffisante dans ce cas (indicateur RD=0). Ce bloc ainsi défini est valable pour la certification tout comme l'authentification.

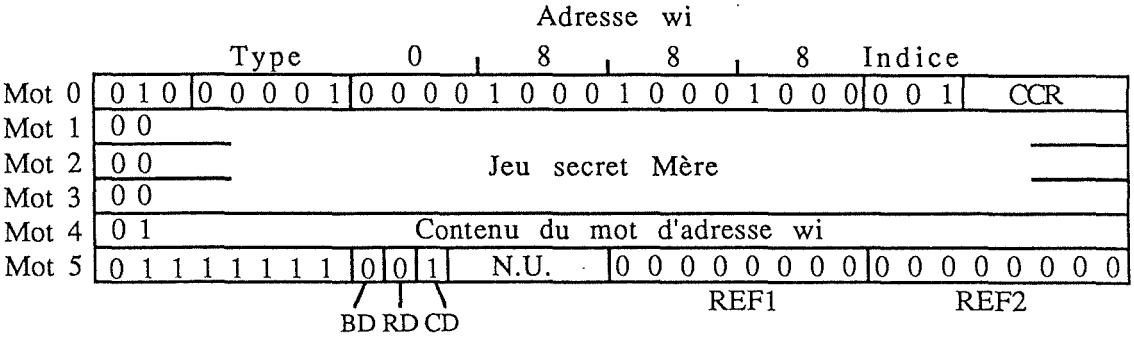


Figure 2.7. : Bloc d'authentification / certification

2.5.3. Le chiffrement

Le chiffrement a pour objectif de protéger l'information contre tout espion. Il s'agit d'un mécanisme qui permet au serveur, tout comme au terminal qui reçoit la carte, de disposer d'une même clé pour une transaction, sans qu'il y ait eu une communication de cette clé. Il faut disposer aussi d'un système cryptographique fonctionnant sur base d'une clé secrète partagée par les deux interlocuteurs.

Pour générer une clé secrète, le principe de la certification est à nouveau mis à contribution, la convention ici étant de faire un certificat sur le numéro de série. Bien entendu, grâce à l'intervention d'un nombre aléatoire dans le calcul du certificat, la clé secrète utilisée pour le chiffrement est différente pour chaque session, ce qui augmente grandement la sécurité du système.

En ce qui concerne la conception du bloc secret, c'est la même chose que le bloc secret de certification, à l'exception près que l'utilisation de la fonction algorithmique est limitée (CD=0) à l'adresse du mot de la zone de fabrication de la carte contenant le numéro de série. De ce fait ce bloc ne permet pas d'établir de certificat. Le résultat du calcul doit être visible puisqu'il s'agit ici de la production d'une clé et non d'un contrôle comme pour le certificat. Ci-dessous, nous vous présentons un bloc conçu pour le chiffrement (figure 2.8.).

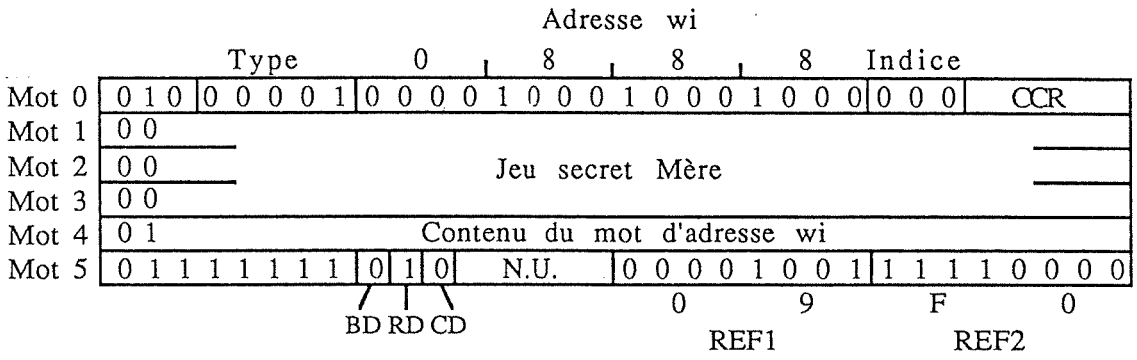


Figure 2.8. : Bloc de clé de chiffrement.

2.5.4. La signature

Le système de signature (électronique) se base sur plusieurs principes [Akl 83] :

- une signature accompagne un message;
- si une information change dans le message, la signature doit être différente;
- si un même message est envoyé deux fois (cas de deux exemplaires) la signature doit être différente (notion de numéro de message);
- il est impossible pour quelqu'un d'autre de fournir une telle signature;
- le signataire ne peut pas renier sa signature.

Au vu de ce qui précède, on peut imaginer une signature en se basant à nouveau sur le principe de certification. Il faut introduire, par exemple en tête du message, un numéro de message. Tout le message est "replié" par une technique similaire à celle utilisée par l'ordre de repli (paragraphe 2.4.8.). On exécute la fonction télépass sur un mot connu de la carte avec comme nombre aléatoire le message replié.

On peut remarquer que les conditions sont bien respectées. En effet, on peut sagement affirmer que la signature n'est plus valable si le message est modifié (cela dépend de la qualité de la technique de repli). Le numéro de message distingue les messages identiques et il n'est pas possible que quelqu'un d'autre produise une bonne signature vu qu'il ne dispose pas du jeu secret de la carte.

La signature peut être vérifiée par un MCS avec un bloc de certification. Il n'est pas possible de contester une bonne signature.

2.5.5. Les autres blocs

Il reste encore une information nécessaire au bon fonctionnement d'un serveur. En effet, ce dernier doit, pour certaines opérations, connaître la clé émetteur primaire (1A) ou secondaire (1B). C'est ainsi que deux nouveaux types de blocs secrets sont constitués pour générer les clés 1A et 1B diversifiées, bien entendu. Il est à noter qu'un bloc spécial est aussi conçu pour les certificats avec replis sur deux octets, sa conception est similaire à celle d'un bloc pour la certification normale.

La constitution d'un bloc secret pour la génération de clé émetteur se base entièrement sur le principe de diversification évoqué au paragraphe 1.8. On doit sélectionner le jeu secret mère **sans** le diversifier et exécuter la fonction algorithmique sur base du numéro de série et du pattern considéré. Le bloc ne provoque pas de diversification (Bd=1), il doit permettre la lecture du résultat (Rd=1), et l'utilisation de la fonction algorithmique est limitée au Wi considéré (CD=0). Ci-dessous, nous vous présentons un bloc conçu pour l'établissement d'une clé 1A (figure 2.9.).

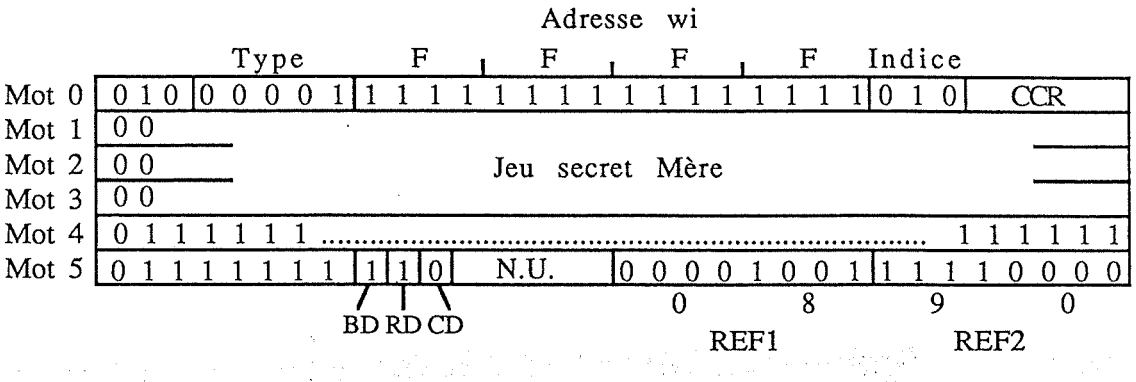


Figure 2.9. : Bloc pour clé émetteur 1A

Chapitre 3

L'ordinateur de sécurité associé[1]

3.1. Introduction

Nous avons vu, dans le premier chapitre, le fonctionnement de la carte. Dans le second, nous avons présenté le module de sécurité qui permet de réaliser des fonctions essentielles au niveau de la sécurité d'une application (authentification, certification, chiffrement).

Deux possibilités se présentent quant à l'utilisation du module de sécurité : la première étant de placer directement le module de contrôle et de sécurité dans le serveur même, la seconde étant de l'intégrer dans un ordinateur spécialisé pour le traitement des problèmes de sécurité de gestion d'un parc de cartes.

C'est la deuxième solution qui a été retenue dans le but de centraliser les problèmes de sécurité. C'est ainsi que la société Bull CP8 a développé en 1983 un Processeur de Sécurité Associé (PSA). Cette version a été élaborée sur un matériel conçu pour cette application. En 1987 une nouvelle implémentation a été décidée, le matériel n'étant plus cette fois un matériel spécifique puisqu'il s'agit d'un Personnal Computer (PC). Cette nouvelle conception est dénommée Ordinateur de Sécurité Associé (OSA). Deux cartes sont ajoutées au PC : une première pour la communication avec le serveur (l'ordinateur qui a besoin des services offerts par les MCS) et une seconde qui recoit les MCS. Cette dernière carte nommée PDS44, qui peut accepter jusqu'à quatre MCS, a été développée par la société Bull CP8. Il est à noter qu'elle peut être présente en double exemplaire dans le PC ce qui permet d'avoir 8 MCS dans un OSA. Nous allons vous présenter la première version de l'OSA que nous avons développée lors du stage effectué à la société Bull CP8.

[1] La formulation de "ordinateur de sécurité associé" (OSA) a été développée selement pour ce mémoire. Toute similitude avec un produit commercialié serait un pur hasard.

3.2. La description générale du processeur de sécurité généralisé

3.2.1. L'objectif de l'OSA

L'objectif de l'OSA est de décharger le serveur du problème du calcul des clés, des certificats, etc. et de centraliser les questions de sécurité en un point. Cet objectif était déjà atteint par le PSA.

L'objectif supplémentaire de l'OSA est, par une structure modulaire, d'accroître la souplesse du processeur de sécurité, souplesse au niveau du protocole de communication avec le serveur (X25, Tokenring, Ethernet,...), de la puissance de traitement et enfin d'offrir une très grande facilité pour l'ajout des traitements nécessaires dans le but de pouvoir supporter la gestion de nouveaux types de cartes.

L'OSA offre une dizaine d'ordres permettant de réaliser les fonctions nécessaires pour la sécurité d'une application. (authentification, certification, chiffrement et production de clé émetteur). Pour des raisons de compatibilité, l'OSA supporte tous les ordres qui étaient disponibles sur le PSA.

3.2.2. Les différentes structures d'un serveur

Trois structures sont envisagées. La première se base sur un serveur qui dispose d'un ordinateur frontal et d'un OSA. Ci-dessous, la figure 3.1 vous présente cette structure.

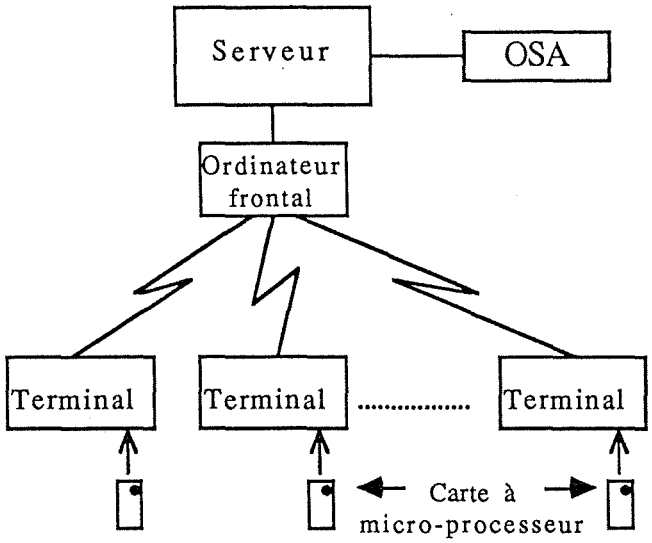


Figure 3.1. : Structure avec ordinateur frontal.

La deuxième structure envisagée est, comme le montre la figure 3.2, identique à la précédente, si ce n'est qu'il n'y a plus d'ordinateur frontal. Cette substitution est totalement transparente à l'OSA; de ce fait les versions du logiciel sont identiques.

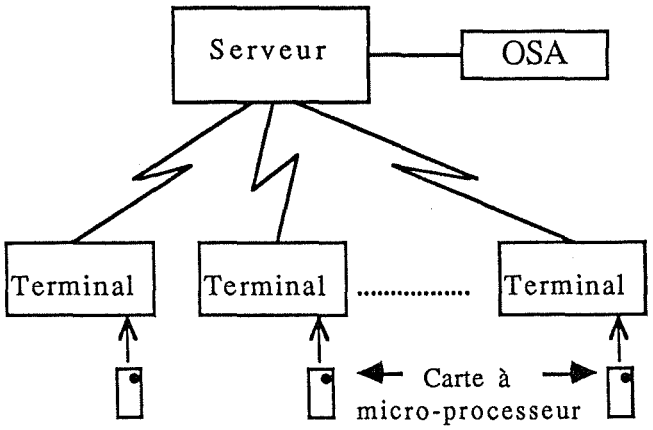


Figure 3.2. : Structure sans ordinateur frontal.

La dernière version consiste en un micro-serveur. Dans cette optique, comme on peut l'observer sur la figure 3.3, le serveur et le PC de l'OSA sont confondus. On remarque que dans cette version, le serveur n'est connecté qu'à

un seul terminal à un moment donné; l'OSA se résume ici à une sorte de boîte à outils fournie au logiciel serveur.

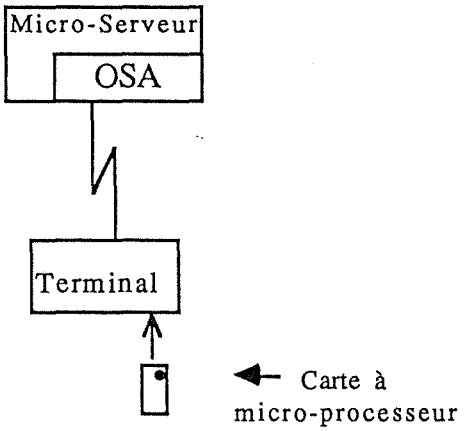


Figure 3.3. : Structure micro - serveur.

3.2.3. La première implémentation

La première implémentation se fait avec un protocole de communication X25 et dans le cadre du traitement des cartes Masque 4, avec une configuration où le serveur est séparé de l'OSA. Cette première implémentation est configurée de telle manière qu'elle supporte le traitement de 50 cartes en parallèle. Il est à noter qu'il s'agit bien d'une première implémentation qui ne sera pas commercialisée. L'OSA comprendra, dans ses versions "commerciales", de nouvelles fonctionnalités.

3.2.4. Le choix du langage de programmation

Le langage C a été choisi comme langage d'implémentation pour sa gestion aisée de la mémoire (allocation, désallocation mémoire, gestion de pointeurs,...), ce qui est essentiel dans le cadre de ce type de logiciel, proche d'un mini système d'exploitation (file d'attente, notion de session,...). Sa facilité de mise en oeuvre (par rapport à l'assembleur) et sa structure rendent le C tout à fait approprié pour construire un logiciel modulaire et qui peut facilement être mis à jour.

3.2.5.Les fonctionnalités de l'OSA

3.2.5.1. La restauration de l'OSA

Cette fonctionnalité provoque une (ré)initialisation complète de l'OSA. Tous les traitements qui étaient toujours en cours sont purement et simplement supprimés. Il est à noter qu'au démarrage de l'OSA une restauration est faite automatiquement.

3.2.5.2. L'initialisation d'une session

L'initialisation d'une session correspond à l'ouverture du dialogue entre le serveur et l'OSA pour le traitement d'une carte. Le serveur communique un numéro de session et les informations caractérisant la carte à traiter (numéro de secret, type de carte, numéro de série,...). Avec ces informations l'OSA calcule la clé de chiffrement et renvoie un nombre aléatoire (le nombre E utilisé dans l'authentification, la certification et le chiffrement).

3.2.5.3. Le chiffrement / déchiffrement

L'OSA se charge de chiffrer ou de déchiffrer des données communiquées. La technique de chiffrement utilisée est un générateur d'octet chiffrant (GOC) initialisé avec la clé de chiffrement calculée lors de l'initialisation de la session et le numéro du message.

3.2.5.4. La demande de clé de chiffrement

L'OSA communique soit la clé de chiffrement de la session en cours, soit une clé calculée sur base de données (caractéristiques d'une carte) communiquées par le serveur dans le cas de messagerie électronique.

3.2.5.5. Le contrôle d'un certificat

Le contrôle d'un certificat peut s'effectuer sur base de données propres à la session mais aussi sur base de données (caractéristiques d'une carte) communiquées, cela dans le cadre d'une messagerie électronique.

3.2.5.6. La demande de clé émetteur

Une clé émetteur fournie par l'OSA est calculée sur base des données propres à la session. La clé est toujours chiffrée. On peut tout aussi bien demander la clé émetteur primaire que secondaire.

3.2.5.7. La remise à zéro du numéro d'octet chiffrant

Cette fonctionnalité permet de (re)synchroniser l'OSA et le terminal recevant la carte usager pour le chiffrement des messages.

3.2.5.8. L'interrogation d'une carte mère

L'interrogation d'une carte mère permet au serveur de connaître l'état d'un MCS et les identifiants des secrets qu'il contient.

3.2.5.9. La fin de session

Cette fonctionnalité provoque la clôture du traitement d'une carte (une session). Ceci permet à l'OSA de libérer l'espace mémoire qui était réservé pour la session spécifiée.

3.2.5.10. La fin de fonctionnement de l'OSA

Cette fonctionnalité permet d'arrêter le fonctionnement de l'OSA. elle n'a de sens que dans le cas où le serveur est séparé de l'OSA, elle n'est donc pas implémenté dans la version micro-serveur.

3.2.5.11. La demande de statistique sur les certificats

Une statistique est effectuée durant le fonctionnement de l'OSA. Elle comptabilise le nombre de certificats faux par rapport au nombre total de certificats effectués. Cette fonctionnalité permet de connaître les valeurs statistiques calculées sur les dernières périodes de fonctionnement de l'OSA (Une période étant la durée de fonctionnement qui sépare deux restaurations de l'OSA).

3.2.6. Le fonctionnement général de l'OSA

Dans ce paragraphe nous allons vous présenter les principes de fonctionnement généraux de l'OSA. A l'aide de la figure 3.4. nous passerons en revue les cinq principaux modules qui composent l'OSA. Toute cette présentation est volontairement simplifiée pour bien montrer le noyau de l'OSA.

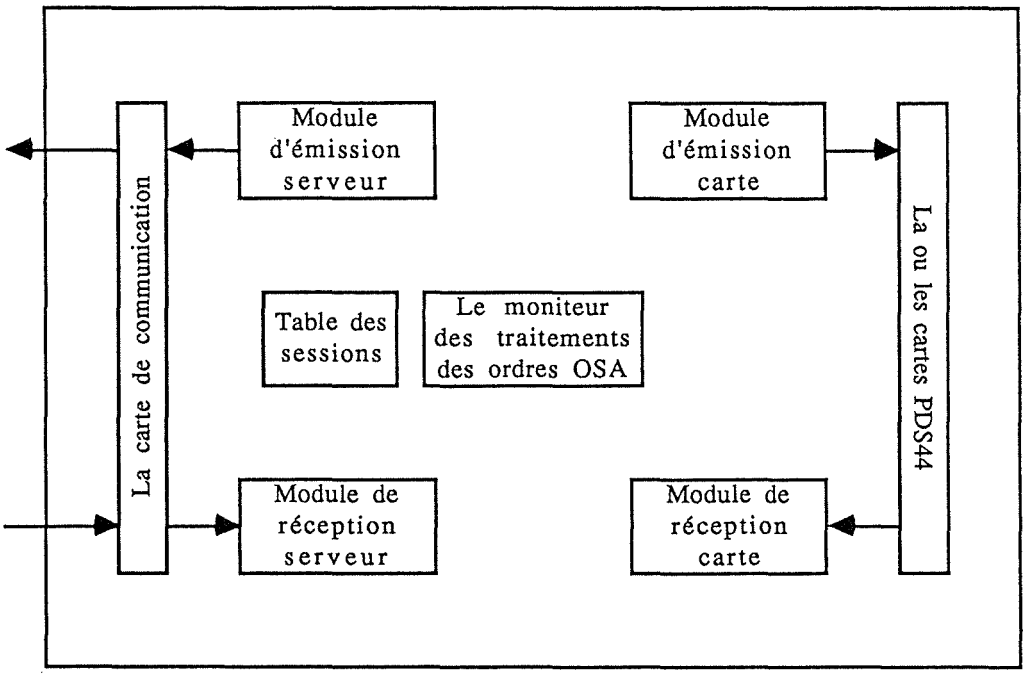


Figure 3.4. : Fonctionnement général de l'OSA.

Quelles sont les particularités de l'OSA qui sont à la base de la conception de cette architecture ? Elles sont au nombre de deux. La première est qu'il y a

deux "côtés" à l'OSA, cela par construction hardware : le "côté" serveur avec la carte de communication et le "côté" MCS avec la ou les cartes Pds44. La seconde est que l'OSA traite plusieurs sessions en parallèle.

A cause des deux "côtés" de l'OSA, on peut observer la double paire Emetteur/Récepteur vers la carte de communication et vers la ou les cartes Pds44. Entre ces deux extrémités on trouve le module de traitement des ordres OSA.

Vu que le OSA doit pouvoir traiter plusieurs sessions en parallèle, il est normal de considérer la notion de session comme la base de l'OSA. Le parallélisme du traitement des sessions est nécessaire par le fait que le temps d'exécution d'une commande MCS est en moyenne de 100 ms.

On peut constater dans les paragraphes précédents que certaines fonctionnalités ne concernent pas le traitement d'une carte, mais d'une session. Deux possibilités se présentaient à nous, à savoir : concevoir un module de traitement suffisamment général pour qu'il puisse traiter aussi bien les fonctionnalités concernant des sessions et les autres fonctionnalités, ou concevoir un module de traitement uniquement pour les fonctionnalités portant sur les sessions, le module de réception serveur traitant les autres fonctionnalités dès leur réception. C'est cette dernière solution qui fut retenue. En effet le but essentiel est de ne faire le parallélisme qu'au niveau des sessions pour bien utiliser les MCS (ressource critique). De plus, les fonctionnalités ne concernant pas les sessions ne sont que des fonctionnalités accessoires et rarement utilisées. Il est aussi indéniable qu'une généralisation du module de traitement des ordres ajouterait une lourdeur handicapante au niveau des performances et de la maintenance du logiciel.

Vu le parallélisme du traitement des sessions il est indispensable d'avoir une structure mémorisant le contexte d'une session : la table de sessions. D'autres structures nécessaires pour le fonctionnement de l'OSA seront présentées dans la description détaillée. Les cinq modules sont activés par un moniteur général qui n'est pas représenté sur la figure.

3.3. Description détaillée de l'OSA

Ci-dessous, la figure 3.5 vous présente une version plus détaillée de la figure 3.4. décrivant les composants de base de l'OSA.

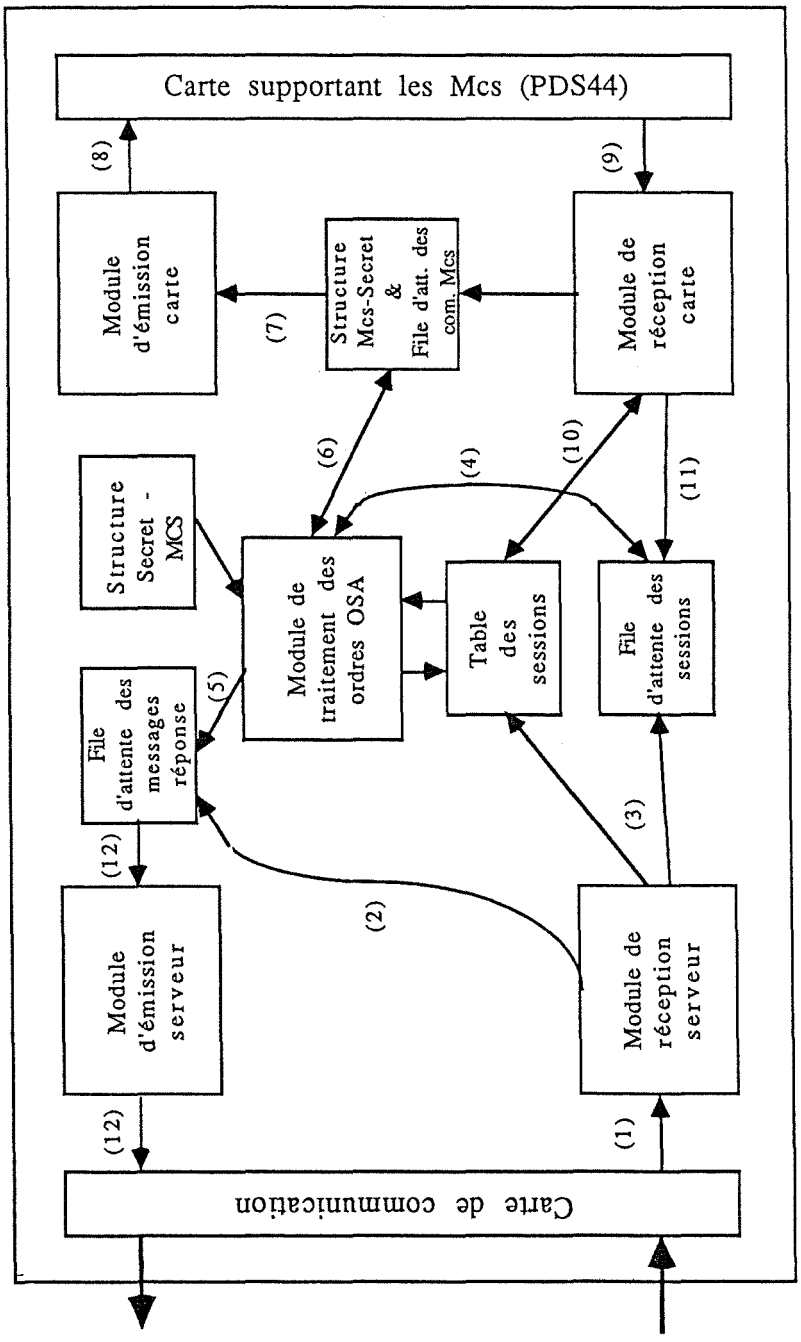


Figure 3.5. : Composants de base du OSA.

En guise d'introduction à la description plus détaillée de l'OSA, nous allons quelque peu vous présenter le processus de traitement d'un ordre dans l'OSA en nous basant sur la figure 3.5.

Tout ordre est d'abord pris en charge par le module de réception serveur (1). Deux possibilités s'offrent ici : soit l'ordre est traité et le message réponse est inséré dans la file d'attente des messages réponses (2) ; soit les informations de la session concernée mémorisées dans la table des sessions sont mises à jour et la session est insérée dans la file d'attente des sessions (3).

Le module de traitement des ordres prend en charge l'ordre concernant la session (4). Soit il le traite, constitue le message réponse et l'insère en file d'attente des messages réponses (5), soit il constitue une suite de commandes[1] et l'insère en file d'attente des commandes (6). On remarque donc que le traitement d'un ordre concernant une session est décomposé en tâches, une à deux selon qu'il y a ou non des commandes à exécuter par un MCS.

Le module d'émission carte se charge, lui, de la sélection d'une commande dans la file d'attente des commandes (7) et la lance sur un MCS (8).

Le module de réception carte réalise la réception de la réponse d'une suite de commandes (9), se charge de la mémoriser dans l'espace mémoire de la session concernée (10) et cette dernière est insérée dans la file d'attente des sessions (11).

Le module d'émission serveur prend en charge l'émission des messages réponses mémorisés dans la file d'attente (12).

Bien entendu cette présentation fut séquentielle et continue pour le traitement d'un ordre. Il est toutefois très important de bien voir que les cinq modules sont relativement indépendants et activés par le moniteur général selon des priorités et le fait qu'ils ont quelque chose à traiter (élément présent dans leur file d'attente ou une réception à effectuer).

[1] Par convention on parlera d'ordre pour les instructions du serveur à destination du PSG et de commande pour les instructions à destination des MCS.

Nous allons dans un premier temps vous présenter les différentes structures évoquées ci-dessus. Ensuite, nous aborderons l'architecture logicielle de l'OSA.

3.3.1. Les différentes structures

3.3.1.1. La table des sessions

La table des sessions possède une entrée par session (dans notre première configuration il y en a 50). Elle mémorise trois types d'information. Il s'agit d'informations système, de données de travail (données conservées pendant l'exécution d'un ordre) et des informations sur le contexte de la session dont la durée de validité est toute la session (tout le traitement d'une carte). Dans le détail, la table des sessions est constituée de :

- un octet de verrouillage signalant que l'occurrence est occupée ou non;
- l'occurrence de la tâche à exécuter pour la réalisation de l'ordre;
- un pointeur vers la zone de travail de la session;
- un pointeur vers la zone de contexte de la session;
- un pointeur vers le message (l'ordre) reçu du serveur.

La zone travail est le moyen de communication entre les tâches ; elle comprend les zones suivantes :

- un code erreur;
- un code traitement qui est, en fait, un paramètre qui détermine l'exécution de la tâche suivante;
- le numéro du MCS sur lequel des commandes ont été lancées par la tâche;
- le Type et l'Indice du bloc qui a été sélectionné pour l'exécution des commandes (nécessaire en cas de reprise sur incident MCS);
- les mots d'état, donnés à la fin de l'exécution des commandes effectuées par le MCS, spécifiant la bonne fin ou non des commandes;
- le résultat fourni par le MCS.

La zone contexte mémorise essentiellement les caractéristiques de la carte que la session traite. Elle comprend :

- le nombre aléatoire E ;
- les paramètres de diversification ;
- le numéro du client ;
- le numéro de série de la carte;

- l'adresse du numéro de série ;
- la clé de chiffrement ;
- le numéro de message chiffrant ;
- le type de la carte (Masque de la carte) ;
- le Type (qui intervient dans l'identifiant d'un bloc secret).

3.3.1.2. La file d'attente des sessions

La file d'attente contient tous les numéros des sessions dont le moniteur des traitements peut commencer ou poursuivre l'exécution.

Le système de liste à été choisi pour implémenter la file d'attente des sessions pour plusieurs raisons. D'abord au niveau de la performance : pas de test, pas de recherche dans une table, une simple consultation de la tête de la liste permet de savoir si une session est en attente. Ensuite, la liste implémente automatiquement une gestion de file d'attente FIFO ce qui garantit la terminaison de chaque session. Toutes les sessions sont sur le même pied d'égalité. Ci-dessous, la figure 3.6 vous présente la structure de liste qui implémente la file d'attente.

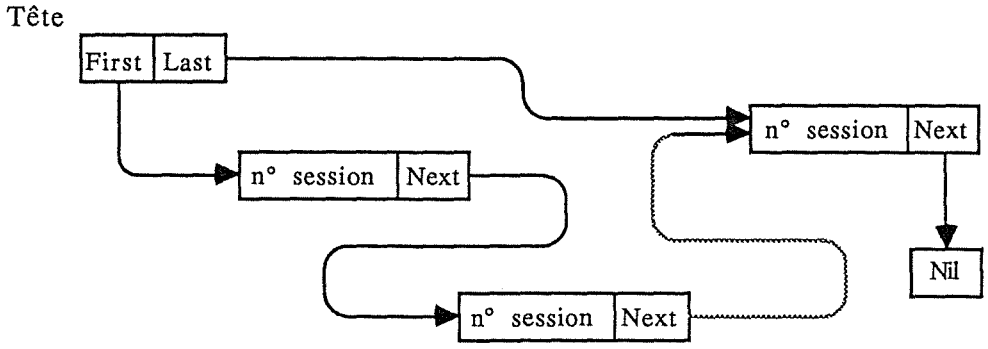


Figure 3.6. : File d'attente des sessions.

3.3.1.3. La file d'attente des messages réponses

La file d'attente des messages réponses a pour objectif la mémorisation de tous les messages réponses en attente d'émission vers le serveur.

Le système de liste a également été choisi pour la file d'attente des messages, exactement pour les mêmes raisons que pour la file d'attente des sessions. Comme on peut le remarquer sur la figure 3.7, les éléments de la liste comprennent seulement un pointeur vers le message ainsi que sa longueur, ceci permettant de gérer de manière aisée des messages de n'importe quelle longueur.

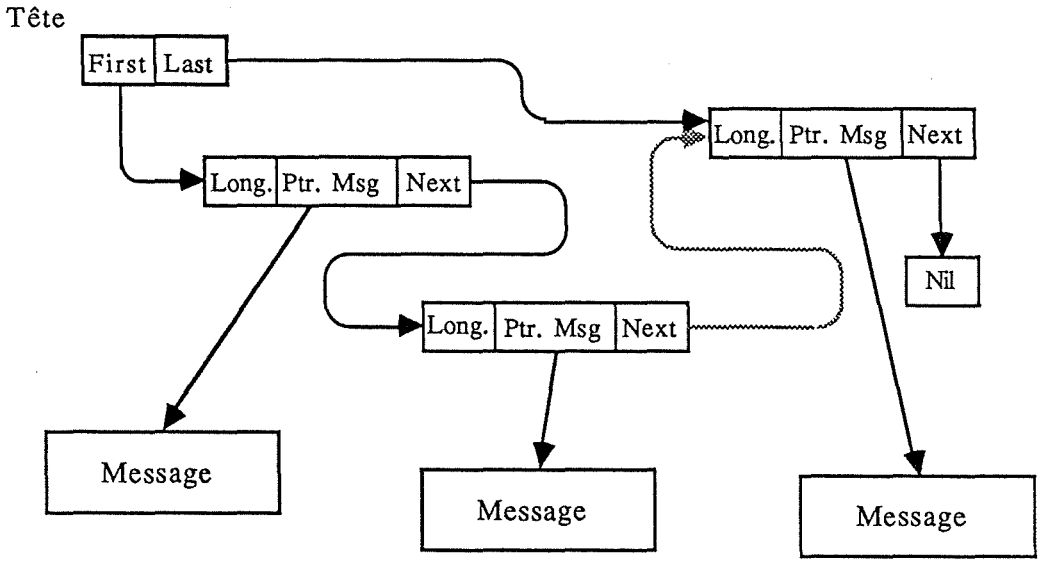


Figure 3.7.: Implémentation de la file d'attente des messages réponses.

3.3.1.4. La structure secret / MCS

La structure secret / MCS permet au moniteur des traitements de connaître quel MCS contient un bloc secret identifié par le couple Type-Indice. Elle se compose, comme on peut l'observer sur la figure 3.8, de deux parties : la première étant une table à 256 entrées, ceci par le fait que l'identifiant d'un bloc secret (Type et Indice) se code sur 8 bits et la seconde d'un ensemble de listes de numéros de MCS. Chaque entrée de la table est la tête de la liste déterminant les numéros de MCS contenant un bloc secret correspondant.

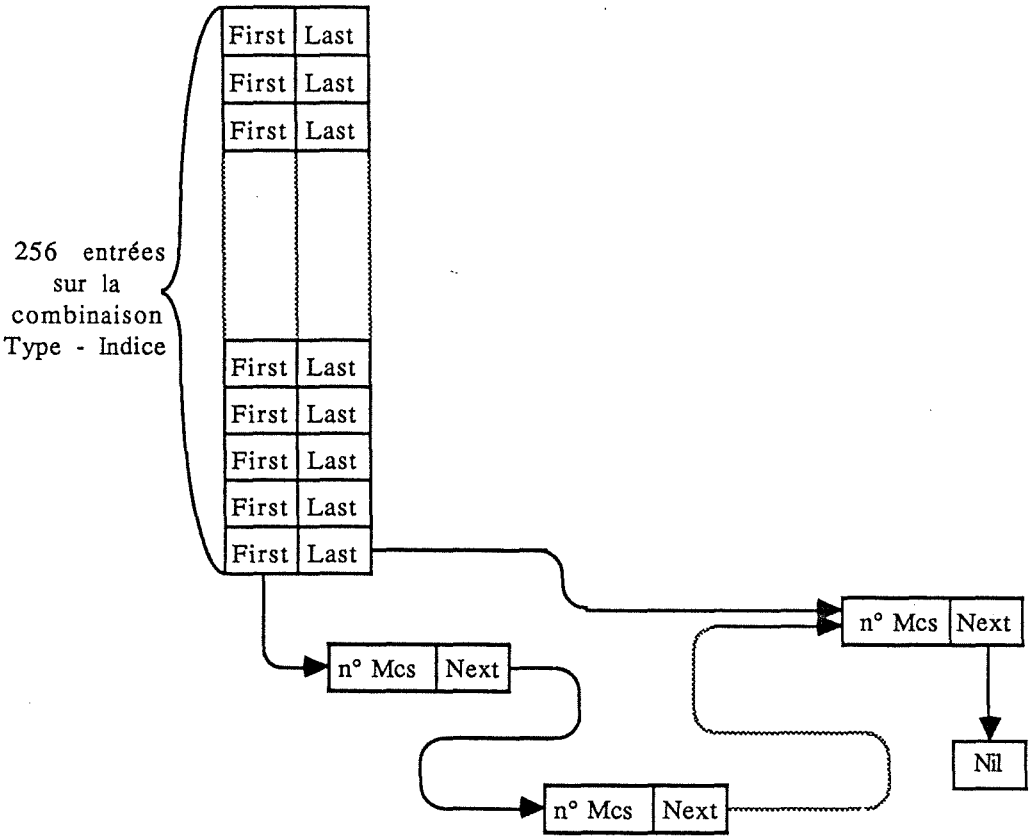


Figure 3.8. : Implémentation de la structure secret / MCS.

3.3.1.5. La structure MCS - secret / commande en attente

La structure MCS - secret / commande en attente a plusieurs objectifs. Elle mémorise les caractéristiques de chaque MCS, la liste des identifiants des blocs secrets de chaque MCS ainsi que la file des commandes en attente sur le MCS. Elle est donc composée de trois éléments : une table et deux listes. La table possède une entrée par MCS et est constituée de :

- un indicateur d'invalidation (signale une panne éventuelle du MCS)
- un indicateur signalant qu'une commande est en cours d'exécution ou non sur le MCS;
- le nombre de commandes en attente sur le MCS;
- le type du MCS (quel masque de carte est-il capable de traiter);
- la tête de la liste des identifiants des blocs secrets contenus dans le MCS;
- la tête de la liste des commandes en attente.

La liste des identifiants des blocs secrets contenus dans le MCS est une simple liste où chaque élément comprend un identifiant d'un bloc secret et un pointeur vers l'élément suivant.

Chaque élément de la liste des commandes en attente comprend :

- un pointeur vers la commande;
- le numéro de la session qui a lancé la commande;
- un indicateur pour signaler un chaînage de commande[1];
- un pointeur vers l'élément suivant.

Ci-dessous la figure 3.9 vous présente un synoptique de la structure MCS - secret / commande.

[1] Pour calculer un certificat, une clé de chiffrement, etc., il faut plusieurs commandes. C'est dans ce cas que la notion de chaînage de commandes prend toute sa signification.

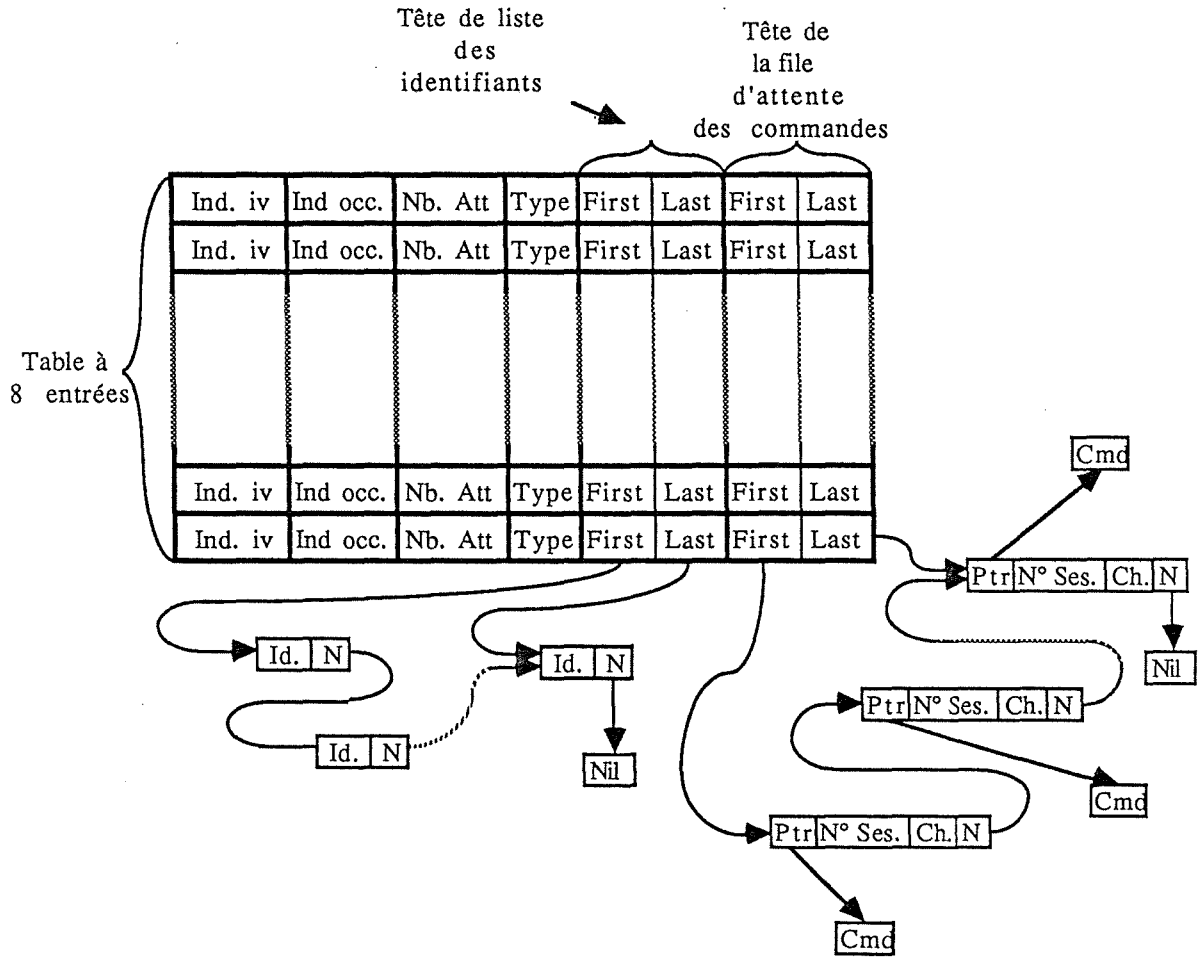


Figure 3.9. : Synoptique de la structure MCS-secret / commande.

3.3.2. L'architecture logicielle de l'OSA

L'OSA a été développé sur base d'une hiérarchie utilise en quatre niveaux :

- le moniteur général;
- les fonctionnalités du logiciel;
- les tâches;
- les utilitaires.

Il est à noter que la présentation de l'OSA faite par la figure 3.5 était de loin incomplète quant à la partie logicielle et avait, comme unique but, la schématisation grossière du mécanisme de fonctionnement de l'OSA. Nous allons maintenant détailler l'objectif de ces différents niveaux.

3.3.2.1. Le moniteur général

Le moniteur général se charge de "donner la main" à une des six principales fonctionnalités du logiciel selon des priorités définies ci-dessous :

1 Réception serveur
2 Emission serveur
3 Réception carte
4 Emission carte
5 Traitement ordre
6 Statistiques

Pour des raisons de sécurité, les statistiques sont exécutées au moins toutes les 200 affectations si la charge n'a pas permis son exécution jusque là.

3.3.2.2. Les fonctionnalités du logiciel

Le deuxième niveau reprend les cinq fonctionnalités traitées par les cinq modules de la figure 3.5 plus trois modules, deux de gestion de l'OSA et le module d'accès au fichier des statistiques :

- initialisation du OSA;
- effacement des données du OSA;
- statistiques;
- réception serveur;
- émission serveur;
- réception carte;
- émission carte;
- traitement ordre.

3.3.2.3. Les tâches

Les tâches constituent une partie ininterruptible du traitement d'un ordre concernant une session. Elles se trouvent au troisième niveau. Bien entendu elles sont utilisées exclusivement par le module de traitement des ordres.

3.3.2.4. Les utilitaires

C'est au quatrième niveau que l'on retrouve un ensemble d'utilitaires principalement destinés aux accès des différentes structures de l'OSA (Recherche d'un MCS, insertion - extraction dans une file d'attente, ...) de même que l'algorithme de chiffrement et de "P1/6sation"(1).

Nous allons maintenant présenter les deux premiers niveaux et ensuite, nous développerons chaque instruction ainsi que sa découpe en tâches. Le quatrième niveau ne sera pas développé étant donné qu'il n'est pas essentiel pour la suite de l'exposé.

3.3.3. Le niveau 1

3.3.3.1. Le moniteur général

Le moniteur général a, comme on l'a déjà mentionné, pour objectif de donner la main aux différents modules du niveau 2 selon certaines priorités. La figure 3.10 présente la structure du moniteur sous forme d'un arbre programmatique(2.) On peut constater qu'il s'agit simplement de deux boucles avec une suite de tests effectués dans un ordre qui respecte les priorités spécifiées précédemment.

(1) La P1/6sation est un mécanisme qui permet de coder un message en étant sur qu'il ne contient aucun caractère de contrôle.
(2) La sémantique de l'arbre programmatique est développée dans l'annexe 1.

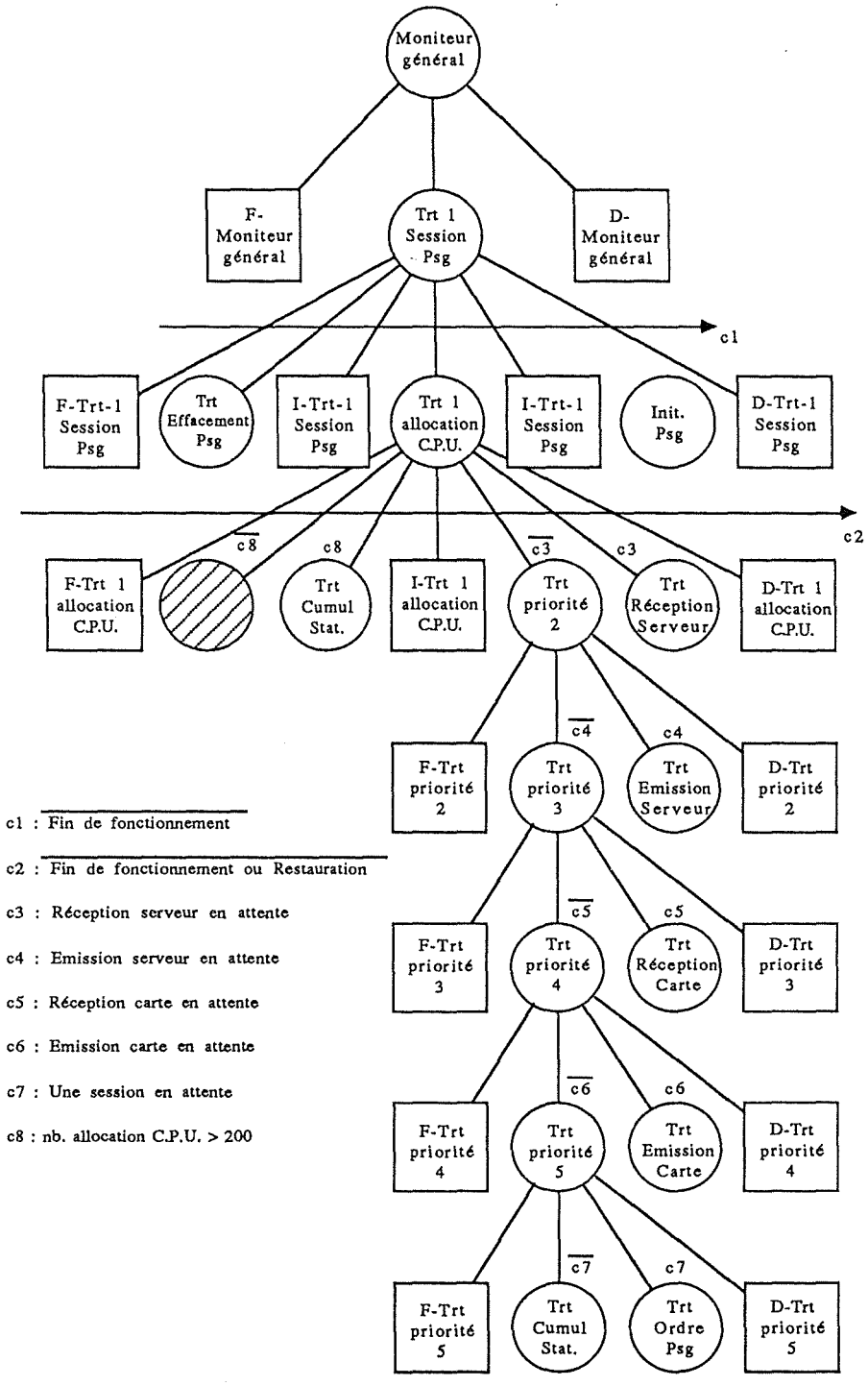


Figure 3.10. : Arbre programmatique du moniteur général.

3.3.4. Le niveau 2

3.3.4.1. L'initialisation de l'OSA

Le module d'initialisation de l'OSA consiste principalement en l'initialisation des différentes tables et files d'attente. C'est aussi à ce moment que l'on écrit un nouvel enregistrement statistique, compteurs initialisés à zéro.

3.3.4.2. L'effacement des données OSA

Le module d'effacement des données OSA réalise la suppression de toutes les données qui ont été créées dynamiquement. Ce module est invoqué lors d'une restauration de l'OSA, cela avant d'effectuer la réinitialisation proprement dite.

3.3.4.3. Les statistiques

Le module de statistique se charge de cumuler les compteurs qui comptabilisent les certificats dans l'enregistrement de la session en cours. Il contrôle que le rapport certificat incorrect sur certificat correct ne dépasse pas un certain seuil fixé. Dans ce cas, le module statistique envoie un message d'alerte au serveur.

3.3.4.4. La réception serveur

Le module de réception serveur prend en charge une partie du protocole de réception (la carte de communication ne prenant en charge que les trois premières couches ISO). Il effectue une partie des contrôles sur les messages. Cette partie est déterminée par le fait que l'on communique un message au module des traitements des ordres OSA par la table des sessions et donc en fonction du numéro de session. Le module de réception serveur réalise le contrôle du numéro de session et, en cas d'erreur, constitue le message réponse au serveur.

Le module de réception serveur se charge aussi du traitement de tous les ordres qui ne concernent pas une session, à savoir :

- la demande de statistique
- la restauration
- interrogation carte mère

En ce qui concerne les ordres destinés à une session, le module de réception serveur effectue la mise à jour de la table des sessions ainsi que la mise en file d'attente de la session.

3.3.4.5. Le module d'émission serveur

Le module d'émission serveur se charge seulement d'extraire le premier message réponse de la file d'attente des messages réponse et de lancer le protocole d'émission.

3.3.4.6. Le module de réception carte

Le module de réception carte se charge de la réception du résultat d'une commande sur un MCS, ce qui comprend aussi bien l'exécution d'un protocole de communication avec la carte PDS44 qu'un premier traitement de la réponse reçue. Deux cas peuvent se présenter selon que la commande était chaînée ou pas.

Dans le cas où la commande était chaînée, seule l'extraction de la file d'attente du MCS concerné est effectuée (seul le résultat de la dernière commande est communiqué à la session).

Dans le cas où la commande n'était pas chaînée (dernière commande d'une chaîne), un contrôle des statuts renvoyés par le MCS est effectué. Si une erreur MCS est survenue, une routine de reprise des commandes encore en attente sur ce MCS est lancée. Dans tous les cas le résultat de la chaîne de commandes est communiqué à la session concernée par une mise à jour de sa zone de travail. La session concernée est introduite dans la file d'attente des sessions.

3.3.4.7. Le module d'émission carte

Le module d'émission carte se charge de sélectionner une commande en attente sur un MCS, de même que de l'exécution de tout le protocole de communication avec la carte PDS44 pour l'envoi de la commande sur le MCS désiré.

3.3.4.8. Le module de traitement des ordres OSA

Le module de traitement des ordres OSA assure le séquencement des différentes tâches de même que certains contrôles (code opération, séquencement des ordres).

Deux cas de figures se présentent. Le premier : quand la session est sélectionnée pour la première fois en vue du traitement de cet ordre. Dans ce cas les contrôles cités ci-dessus sont effectués et, selon leur résultat, la première tâche est lancée ou non. Le second cas se produit lorsque la session est sélectionnée pour la deuxième fois pour le traitement de la commande ; seule la tâche est alors lancée.

Le module de traitement des ordres se base sur une table à deux dimensions pour déterminer la tâche à exécuter. Les deux dimensions étant basées sur le code opération de l'ordre et sur l'occurrence de la tâche à effectuer, l'élément ainsi sélectionné spécifie un numéro la tâche à effectuer.

3.3.5. Les ordres et leur traitement

Tous les ordres ont une même structure de base. Comme on peut l'observer sur la figure 3.11 la structure de base est composée de quatre parties : le code opération sur un octet, le numéro de session toujours sur un octet (pour certains ordres, il est présent mais pas utilisé), une zone de deux octets déterminant la longueur de la zone donnée.

Code opération	Numéro de session	Longueur 3X	Longueur 3Y	Champ de données

Figure 3.11. : Structure de base d'un ordre OSA.

La longueur est sous la forme 3X-3Y, ceci pour des raisons de protocole lors de l'élaboration du PSA ; en effet, avec ce système on peut garantir qu'il n'y a pas de caractère de contrôle, la longueur étant bien entendu XY sur un octet.

Le champ de données est évidemment présent seulement si le champ longueur est différent de 3030. Dans le PSA, les données étaient automatiquement "P1/6sées" ; dans l'OSA, vu que l'on utilise d'autres protocoles de communication qui peuvent être transparents, les données seront "P1/6sée" ou non selon la valeur d'un paramètre donné à la restauration. Par défaut, elles ne sont pas "P1/6sées".

Certains ordres utilisent plusieurs champs de données. Dans ce cas, le couple longueur/champ de données est répété, cela jusqu'à trois fois au plus.

Pour chaque message reçu, l'OSA renvoie un message réponse qui a exactement la même structure que les messages en entrée, excepté le code opération qui est remplacé par un code d'erreur.

Nous allons maintenant vous présenter les ordres disponibles sur l'OSA ainsi que la forme du message réponse en cas de bon fonctionnement (code d'erreur = 40). La liste des codes d'erreurs est présentée en annexe 2.

3.3.5.1. L'ordre de restauration

L'ordre de restauration permet au serveur d'initialiser ou réinitialiser complètement l'OSA. En outre, selon le code opération, il spécifie si l'on "P1/6se" ou non le champ de données des ordres suivants (38 entraîne la "P1/6sation"). La figure 3.12 nous montre la structure de l'ordre de restauration.

32 ou 38	N.U.	30 | 30

Figure 3.12 : Ordre de restauration.

Vu que l'ordre de restauration ne concerne pas une session, il est pris en charge par le module de réception serveur. On peut toutefois dire que le traitement de restauration proprement dit se réalise en grande partie par des invocations du moniteur général. En effet, la restauration est traitée par les modules d'initialisation OSA et d'effacement des données OSA, le module de réception serveur se bornant à positionner l'indicateur de restauration(1) et à constituer le message réponse (présenté par la figure 3.13) qui doit être envoyé avant la restauration proprement dite.

40	N.U.	30 | 30

Figure 3.13. : Message réponse de la restauration.

3.3.5.2. L'ordre d'initialisation 1

L'ordre d'initialisation 1 est la première partie de l'ouverture d'une session (traitement d'une carte). La figure 3.14 nous décrit sa structure.

30	Numéro de session	30 | 30

Figure 3.14. : Ordre d'initialisation 1.

Cet ordre correspond à la demande d'ouverture d'une session donnée. L'OSA renvoie le nombre aléatoire E. Pour rappel, le traitement du contrôle du numéro de session est fait par le module de réception serveur. Une tâche se charge du tirage d'un nombre aléatoire, de sa mémorisation dans la zone

(1) Voir l'arbre programmatique du moniteur général au paragraphe 3.3.3.1.

contexte de la session ainsi que de la constitution du message réponse présenté par le figure 3.15.

40	Numéro de session	30 | 36	Le nombre aléatoire E (48 bits)

Figure 3.15. : Message réponse de l'initialisation 1.

3.5.5.3. L'ordre d'initialisation 2

L'ordre d'initialisation 2 permet au serveur de communiquer à l'OSA les caractéristiques de la carte qui sera traitée. C'est ainsi que l'on peut observer, dans le champ de données de l'ordre (figure 3.16) la présence du numéro de client, du numéro de série, de l'adresse du numéro de série mais aussi NS, le numéro de secret utilisé (Le Type, partie de l'identifiant du bloc secret), TC déterminant la technique de diversification utilisée (sur base du numéro de client ou du numéro de série), TAP et Num-logiciel ne sont pas pris en compte par l'OSA.

34	Num. Ses.	31 | 32	TAP	NC1	NS	TC	Num Client 2	Numéro de série	Adr. série	Num. Logic

Figure 3.16. : Ordre d'initialistation 2.

Quatre traitements doivent être effectués pour cet ordre. Tout d'abord, la mémorisation des informations de la carte dans la zone contexte de la session, ensuite le calcul de la clé de chiffrement, sa mémorisation dans la zone contexte de la session et finalement la constitution du message réponse qui comprend, comme le montre la figure 3.17, la date et l'heure.

40	Num. Ses.	30 | 3C	TAP	A A M M J J H H m m

Figure 3.17. : Message réponse de l'initialisation 2.

Comme il y a utilisation d'un MCS pour le calcul de la clé de chiffrement, deux tâches sont nécessaires. La première se charge des premières mémorisations et de la constitution des commandes, la seconde du traitement du résultat du MCS, de la mémorisation de la clé de chiffrement et de la constitution du message réponse.

3.5.5.4. L'ordre de chiffrement / déchiffrement

L'ordre de chiffrement / déchiffrement permet, comme son nom l'indique de chiffrer ou de déchiffrer. Le système de chiffrement choisi est un générateur d'octets chiffrants (GOC). Les octets générés sont appliqués au message par une opération XOR. De ce fait il n'y a aucune différence au niveau des traitements entre le chiffrement et le déchiffrement. La figure 3.18 donne la structure de l'ordre.

35	Numéro de session	3X | 3Y	Données à chiffrer ou déchiffrer

Figure 3.18. : Ordre de chiffrement / déchiffrement.

Pour ce traitement une seule tâche suffit puisqu'il n'y a pas d'utilisation de MCS. Cette tâche se charge du chiffrement et de la constitution du message réponse présenté ci-dessous par la figure 3.19. A chaque chiffrement / déchiffrement, le numéro d'octet chiffrant mémorisé dans la zone contexte de la session concernée est augmenté d'une unité. Il est à remarquer que la dénomination "numéro d'octet chiffrant" est quelque peu erronée puisqu'il s'agit de la mémorisation d'un numéro de message !

40	Numéro de session	3X | 3Y	Données chiffrées ou déchiffrées

Figure 3.19 : Message réponse du chiffrement / déchiffrement.

3.5.5.5. L'ordre de demande de clé de chiffrement sur données internes

L'ordre de demande de clé de chiffrement sur données internes permet au serveur de connaître la clé de chiffrement. Il est utilisé dans le cas où l'algorithme de chiffrement employé pour une application n'est pas le GOC implémenté dans l'OSA mais un autre algorithme qui est exécuté par le serveur. La structure de l'ordre est présentée par la figure 3.20. Vu qu'il s'agit d'une demande il n'y a évidemment pas de champ de données à cette instruction.

3D	Numéro de session	30 | 30

Figure 3.20. : Ordre de demande de clé de chiffrement sur données internes.

Le traitement est effectué par une seule tâche qui se charge simplement de constituer le message réponse à partir de la clé de chiffrement mémorisée dans la zone contexte de la session. Ci-dessous la figure 3.21 vous donne la structure du message réponse.

40	Numéro de session	30 | 38	Clé de chiffrement (64 bits)

Figure 3.21. : Message réponse de la demande de clé de chiffrement sur données internes.

3.5.5.6. L'ordre de demande de clé de chiffrement sur données externes

L'ordre de demande de clé de chiffrement sur données externes permet au serveur de connaître la clé de chiffrement utilisée par une autre carte lors d'un chiffrement effectué par cette dernière. Cet ordre est utilisé dans le cas de messagerie électronique. En effet, si un utilisateur a mémorisé dans une boîte aux lettres électronique un message qu'il a chiffré grâce à la clé de

chiffrement de sa carte, il faut que le destinataire du message puisse le déchiffrer. Par conséquent, il est nécessaire de pouvoir connaître la clé de chiffrement qui a été utilisée. Bien entendu, comme le montre la figure 3.22, les données de diversification sont fournies sous le même format que celui qui avait été utilisé pour l'ordre d'initialistion 2. Toutefois, deux formats sont disponibles : l'un donnant toutes les données de diversification, l'autre donnant seulement le nombre E. Dans ce cas, la clé de chiffrement est calculée avec les paramètres de diversification de la session et bien entendu le nombre E communiqué.

3E	Numéro de session	30	30	30	36	Le nombre E		

3E	Numéro de session	31	32	Format identique à la zone donnée de l'Init. 2		30	36	E

Figure 3.22. : Ordre de demande de clé de chiffrement sur données externes.

Pour le traitement de cet ordre, deux tâches sont nécessaires vu le calcul de la clé de chiffrement sur un MCS. La première tâche se charge de la constitution de la chaîne de commandes pour le calcul de la clé de chiffrement tandis que la seconde traite le résultat de la chaîne de commandes et constitue le message réponse présenté par la figure 3.23.

40	Numéro de session	30	38	Clé de chiffrement (64 bits)

Figure 3.23. : Message réponse de la demande de clé de chiffrement sur données externes.

3.5.5.7. L'ordre de certification et/ou déchiffrement en lecture ou en écriture

L'ordre de certification et/ou chiffrement en lecture ou en écriture permet de composer différentes opérations qui sont la lecture ou l'écriture, le contrôle du certificat et le chiffrement. Quatre combinaisons sont ainsi disponibles :

- lecture chiffrée sans certification ;
- lecture non chiffrée avec certification ;
- lecture chiffrée avec certification ;
- écriture non chiffrée avec certification.

Ci-dessous, la figure 3.24 vous présente les trois structures que peut prendre l'ordre. Il n'y a que trois structures étant donné que la deuxième est valable pour les ordres de lecture avec certification chiffrée et non chiffrée. Différentes informations sont présentes : le certificat à vérifier ainsi que les mots sur lesquels il porte mais aussi NM, AM, AZ et AE qui sont des informations qui permettent d'associer à chaque mot l'adresse où ils sont mémorisés dans la carte (nécessaire vu que l'adresse de stockage du mot intervient dans le calcul du certificat).

3 A	Num. Ses.	3X | 3Y	Type	NM	AM	Mots

3 A	Num. Ses.	3X | 3Y	Type	NM	AM	Certificat	AZ	AE	Mots

3 A	Num. Ses.	3X | 3Y	Type	NM	AM	Certificat	AZ	AE	3X | 3Y	Mots

Figure 3.24. : Différentes structures de l'ordre de certification et/ou déchiffrement en lecture ou écriture.

Bien entendu, deux tâches sont nécessaires vu l'utilisation d'un MCS pour le calcul du certificat. La première se charge d'identifier le traitement à effectuer et de la constitution éventuelle d'une chaîne de commandes ; la

seconde effectue l'analyse du résultat de la chaîne des commandes ainsi que la constitution du message réponse.

Dans le cas où il s'agit d'une lecture chiffrée, les deux tâches s'enchaînent sans exécution de commandes. Ci-dessous, la figure 3.25. vous donne les différentes structures des messages réponses.

40	Num. Ses.	3X 3Y		NM	AM	Mots

40	Num. Ses.	3X 3Y	Etat Cert.	NM	AM	Mots

40	Num. Ses.	3X 3Y	Etat Cert.	NM	AM

Figure 3.25. : Les différentes structures du message réponse de certification et/ou déchiffrement en lecture ou écriture.

3.5.5.8. Contrôle de certificat sur données externes

Le contrôle de certificat sur des données externes permet de contrôler, comme son nom l'indique, un certificat mais ceci sur base des paramètres de diversification d'une autre carte. Ceci est évidemment utilisé dans le cas de messagerie électronique.

Comme le montre la figure 3.26, la structure de l'ordre comprend trois parties dont la première est identique à la structure de l'ordre de certification dans le cadre d'une lecture chiffrée ou non et certifiée(1). La seconde étant le nombre aléatoire E. Sa présence est facultative. S'il est absent, le nombre E de la session est utilisé. Le même principe est appliqué pour la troisième partie qui donne les caractéristiques de la carte, cela par un format identique à la zone donnée de l'ordre d'initialisation 2.

(1) Voir au paragraphe 3.5.5.7.

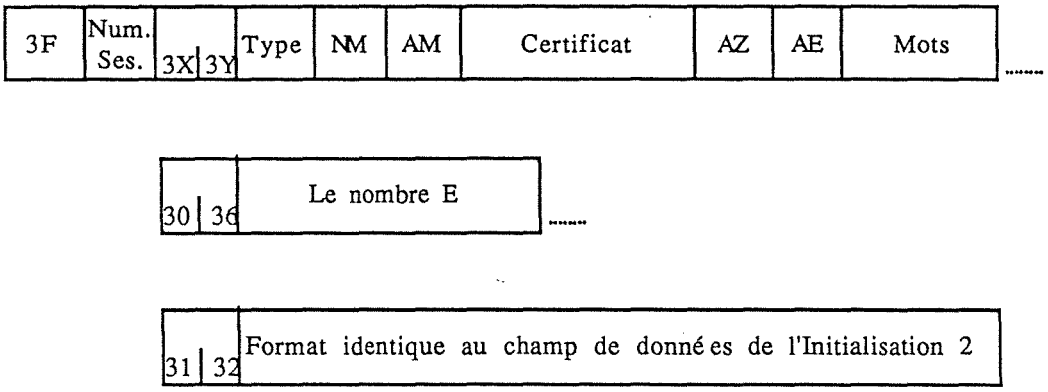


Figure 3.26. : Ordre de certification sur données externes.

Ici aussi deux tâches se chargent du traitement de cet ordre : la première constituant la chaîne de commandes et la deuxième composant le message réponse présenté par la figure 3.27, cela en fonction du résultat des commandes.

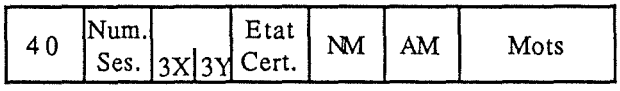


Figure 3.27 : Message réponse de la certification sur données externes.

3.5.5.9. L'ordre de contrôle de certificat avec repli sur deux octets

L'ordre de contrôle de certificat sur deux octets est un cas particulier du contrôle de certificat. Le certificat dans ce cas ne peut porter que sur un seul mot (4 octets) et le résultat du certificat est "replié"(1) sur deux octets. Comme on peut le remarquer sur la figure 3.28, la structure de l'ordre respecte celle des certifications précédentes pour une raison de standardisation.

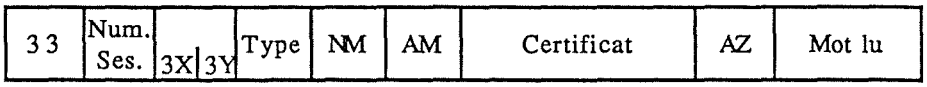


Figure 3.28. : Ordre de contrôle de certificat avec repli sur deux octets.

(1) Voir l'ordre de repli de résultat sur le MCS dans le paragraphe 2.4.8.

Le traitement est effectué par deux tâches où la première se charge de la constitution de la chaîne de commandes et la seconde de l'établissement du message réponse (voir figure 3.29) en fonction du résultat des commandes.

40	Num. Ses.	3X | 3Y	Etat Cert.	NM	AM	Mot lu

Figure 3.29. : Message réponse du contrôle de certificat avec repli sur deux octets.

3.5.5.10. L'ordre de demande de clé émetteur

L'ordre de demande de clé émetteur est nécessaire pour que le serveur puisse écrire dans certaines zones protégées de la mémoire de la carte. Ci-dessous, la figure 3.30 vous donne la structure de l'ordre.

3B ou 3C	Numéro de session	30 | 30

Figure 3.30. : Ordre de demande de clé émetteur.

Deux tâches sont nécessaires ici, étant donné le calcul de la clé émetteur par un MCS. La première se charge, comme d'habitude, de la constitution de la chaîne de commandes, tandis que la seconde constitue le message réponse en fonction du résultat des commandes. La clé ainsi produite est toujours chiffrée. Ci-dessous, la figure 3.31 donne la structure du message réponse.

40	Numéro de session	30 | 38	Clé émetteur 1A ou 1B

Figure 3.31. : Message réponse de la demande de clé émetteur.

3.5.5.11. L'ordre de remise à zéro du numéro d'octet chiffrant

L'ordre de remise à zéro du numéro d'octet chiffrant permet de resynchroniser l'OSA et la carte. La structure de l'ordre est présentée ci-dessous par la figure 3.32.

37	Numéro de session	30	30

Figure 3.32. : Ordre de remise à zéro du numéro d'octet chiffrant.

Cet ordre ne nécessite qu'une seule tâche qui se charge de la mise à zéro du numéro d'octet chiffrant mémorisé dans la zone contexte de la session. La figure 3.33 vous présente la structure du message réponse.

40	Numéro de session	30	30

Figure 3.33. : Message réponse de la remise à zéro du numéro d'octet chiffrant.

3.5.5.12. L'ordre d'interrogation d'une carte mère MCS

L'ordre d'interrogation d'une carte mère MCS permet au serveur de connaître l'état d'un MCS ainsi que tout les identifiants des blocs secrets qu'il contient. Comme on peut le constater sur la figure 3.34 l'ordre spécifie un MCS par son numéro (de 0 à 7).

6F	N.U.	30	31	N

Figure 3.34. : Ordre d'interrogation d'une carte mère MCS.

Cet ordre ne concerne pas une session, il est donc traité directement par le module de réception serveur. Celui-ci se charge donc de la consultation de

la structure MCS - secret / commande en attente, cette dernière permettant l'établissement du message réponse dont la structure est présentée ci-dessous par la figure 3.35.

40	N.U.	3X | 3Y	T	Id. Bloc Secret	.....	Id. Bloc Secret

40	N.U.	30 | 31	E

Figure 3.35. : Message réponse de l'interrogation d'une carte mère MCS.

3.5.5.13. L'ordre de fin de session

L'ordre de fin de session signale la fin du traitement d'une carte. La structure de l'ordre, relativement simple, est présentée par la figure 3.36. En effet, seul le numéro de la session suffit pour clôturer une session.

39	Numéro de session	3X	3Y

Figure 3.36. : Ordre de fin de session.

Pour cet ordre une seule tâche est nécessaire. Elle se charge d'une réinitialisation de l'élément de la table des sessions et de la libération de la mémoire occupée par la zone contexte. Le message réponse présenté par la figure 3.37 est constitué.

40	Numéro de session	3X	3Y

figure 3.37. : Message réponse de la fin de session.

3.5.5.14. L'ordre de lecture du fichier statistique

L'ordre de lecture du fichier statistique permet au serveur de connaître les statistiques effectuées sur les certificats. Cet ordre permet aussi de supprimer le fichier après sa communication. La figure 3.38 montre la structure de l'ordre.

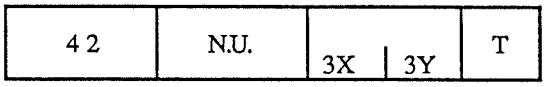


Figure 3.38. : L'ordre de lecture du fichier statistique.

La réalisation de cet ordre incombe au module de réception serveur vu qu'il ne concerne pas une session. Une suite de messages est constituée, chacun comprenant deux enregistrements du fichier statistique (sauf éventuellement le dernier). Selon le paramètre T le fichier est supprimé ou non. La figure 3.39 présente la structure du message réponse.

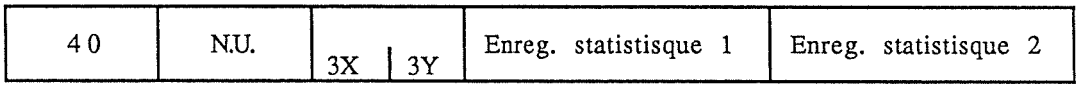


Figure 3.39. : Message réponse de la lecture du fichier statistique.

3.5.5.15. L'ordre de fin de session OSA

Cet ordre permet d'arrêter le fonctionnement de l'OSA. Tout comme dans la restauration, tous les traitements en cours sont purement et simplement supprimés. La figure 3.40 présente la structure de l'ordre.

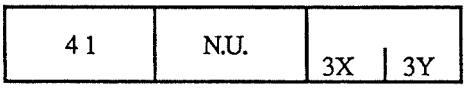


Figure 3.40. : Ordre de fin de session OSA.

Cet ordre est pris en charge par le module de réception serveur. Ce dernier met à jour un indicateur qui provoque la fin du fonctionnement au niveau du moniteur général(1). Il constitue le message réponse présenté par la figure 3.41 et provoque son envoi immédiat (l'insertion se fait au début de la file d'attente).

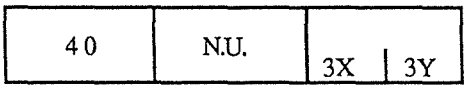


Figure 3.41. : Message réponse de la fin de session OSA.

Ceci termine la présentation de l'OSA. Nous allons étudier cette solution dans les chapitres suivants du point de vue de la sécurité et nous développerons de nouveaux concepts comblant certaines faiblesses de cette implémentation.

(1) Voir la structure du moniteur général au paragraphe 3.3.3.1.

Chapitre 4

Etude de la sécurité de la solution OSA

4.1. Introduction

Nous allons étudier la sécurité offerte par une application basée sur un serveur avec ou sans ordinateur frontal. Nous avions déjà abordé ce sujet au paragraphe 3.2.2. du chapitre précédent. Ci-dessous, la figure 4.1. reprend cette structure sans ordinateur frontal, celle que nous utiliserons, étant donné que la présence d'un ordinateur frontal ne modifie pas le problème de la sécurité.

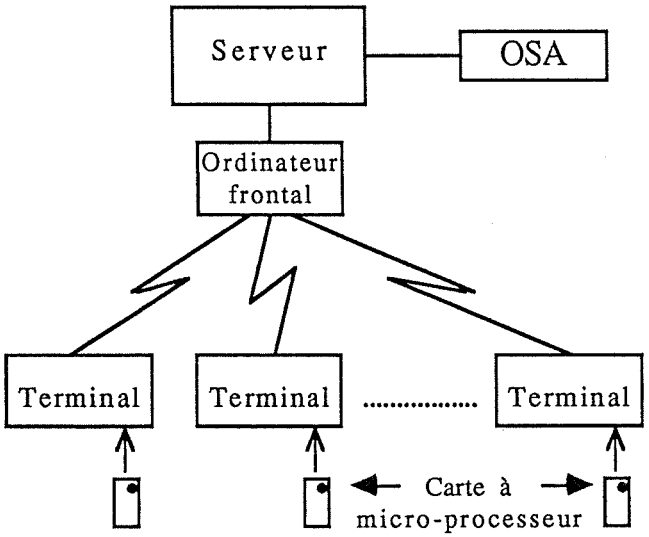


Figure 4.1. : Structure d'un serveur sans ordinateur frontal.

Il est à noter que l'étude qui suit fait totalement abstraction de tout problème technique (taille mémoire des composants, ...), ce qui sera aussi le cas pour le chapitre suivant où nous exposerons quelques solutions aux problèmes soulevés dans ce chapitre.

4.2. La carte à micro-processeur

4.2.1. Introduction

Comme on peut le remarquer sur la figure 4.2., le premier élément de la configuration que nous étudierons est la carte à micro-processeur.

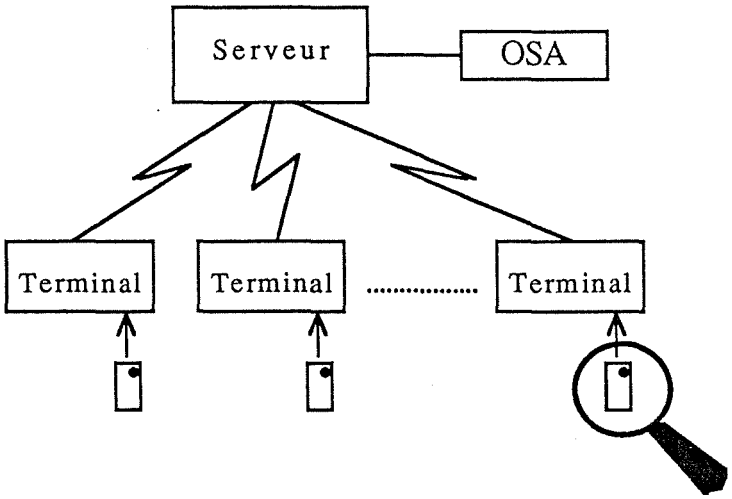


Figure 4.2. : Premier élément : la carte à micro-processeur.

Qu'est-ce que la carte à micro-processeur ? Nous l'avons présentée dans le premier chapitre. Toutefois, il est important de bien souligner ce qu'apporte la carte à micro-processeur.

Dans un premier temps, on remarque que la carte à micro-processeur est un concept d'un support d'informations; dans un second temps, il y a le masque de la carte qui utilise les caractéristiques du support. Nous allons quelque peu détailler ces deux premiers maillons de la "chaîne" de sécurité.

4.2.2. Le concept de support d'informations

Le concept de support d'informations et la structure hardware de la carte présentés dans le premier chapitre par la figure 1.2 sont basés sur un micro-

processeur qui, piloté par un programme mémorisé en ROM, contrôle l'accès à une mémoire EPROM. C'est là l'idée maîtresse de la carte à micro-processeur.

Que peut-on en dire au niveau de la sécurité ? Le principe du contrôle d'accès aux données par un micro-processeur repose sur une caractéristique technique de la technologie utilisée : il est impossible d'accéder par voie physique directe aux données, ou autrement dit, il est impossible d'accéder aux données sans "passer" par le micro-processeur.

4.2.3. Le masque de la carte

4.2.3.1. Introduction

Le masque de la carte, (le programme mémorisé dans la ROM qui pilote le micro-processeur) est un élément très important pour la sécurité. En effet, si ce masque est mal conçu ou mal implémenté cela signifierait qu'il serait possible à un "attaquant" d'utiliser une "lacune" du masque pour accéder à des données pour lesquelles il ne possède pas les droits d'accès requis. Nous avions présenté le masque 4, nous allons reprendre son jeu d'instructions ainsi que la découpe en zone de la mémoire EPROM. Ceci constitue une première approche pour l'étude de la sécurité d'un masque.

Une seconde approche, non moins importante, consiste à vérifier l'implémentation (la programmation) du masque ainsi qe l'étude de l'efficacité de l'algorithme cryptographique utilisé : Télépass. Cette seconde approche ne sera pas abordée , ceci relevant d'une étude plus technique dépassant le cadre de ce mémoire.

Nous allons d'abord passer en revue les ordres disponibles dans le masque 4, ensuite nous examinerons les différents types de zones offertes.

4.2.3.2. Les ordres du masque 4

Le premier ordre est celui de remise à zéro (RAZ) de la carte. Cet ordre hardware a pour seul but l'initialisation du circuit ; la carte renvoie un train

d'octets caractérisant son état. L'action de cette instruction étant limitée, elle n'implique pas de discussion au niveau de la sécurité.

Le second ordre, la lecture est beaucoup plus concernée par les problèmes de sécurité. Vu que sa fonctionnalité est de permettre la lecture d'informations de la mémoire EPROM, il est important qu'au moment de l'exécution de l'ordre, les contrôles des droits d'accès à la zone adressée soient effectués (en partie effectués par l'ordre de validation de clé en lecture), ceci selon les règles d'accès présentées au paragraphe 1.6. du premier chapitre. On peut toutefois constater une lacune au niveau de cet ordre : il livre les données telles quelles, sans chiffrement, brutes et quelle que soit la zone de laquelle ces données proviennent. Si cela n'est pas gênant pour la zone de lecture, cela l'est pour la zone confidentielle et aussi pour la zone de transaction quand l'indicateur de protection en lecture est positionné.

Les mêmes remarques peuvent être faites pour l'ordre d'écriture. Des améliorations pour la présentation des données seront étudiées dans le chapitre suivant.

L'ordre de présentation de clé présente toujours la même lacune : les clés ne sont pas chiffrées. Si la clé porteur est difficilement chiffrable, (cette opération devrait être à charge du porteur), il n'en est pas de même pour la clé émetteur, il n'y a pas de difficulté pour qu'elle soit chiffrée.

On peut comparer la présentation de la clé émetteur avec le principe du calcul d'un certificat ou le mécanisme d'authentification. En effet, on a pris soin pour ces deux traitements que leur message respectif soit différent à chaque fois et cela pour une même carte ou une même donnée à certifier ; le but étant d'empêcher un espion de pouvoir "s'authentifier" ou de produire un certificat simplement en réutilisant le message intercepté. On peut, à juste titre, considérer que la clé émetteur est une donnée confidentielle et qu'il serait intéressant de la protéger en la rendant différente à chaque session. Par conséquent si elle est interceptée, elle ne sera, de toute façon, d'aucune utilité.

L'ordre de validation de clé en lecture permet de contrôler l'accès de la mémoire en lecture. Toutefois, on peut se demander s'il est totalement nécessaire. En effet, une première partie du traitement peut être effectuée par l'ordre de présentation de clé et le reste par l'ordre de lecture.

L'ordre de validation en écriture permet de positionner le bit V d'un mot. Il est important, ici, de séparer l'écriture de la validation, cela pour permettre une vérification de l'écriture avant de la valider. On pourrait toutefois considérer que cette vérification se fait automatiquement par la carte au moment de l'écriture et le bit V serait automatiquement positionné si la vérification était positive.

L'ordre de mise en oeuvre de la fonction algorithmique est un point essentiel de la sécurité de la carte à micro-processeur. La fonction cryptographique utilisée, Télépass, est de type public à clé secrète, ceci bien qu'elle ne soit pas publiée. On doit donc étudier la sécurité qu'elle offre en supposant qu'elle peut-être connue par un attaquant. Malgré ce fait, il est impossible pour un attaquant d'utiliser la fonction Télépass étant donné qu'il ne connaît pas le jeu secret mémorisé dans la carte. Ceci est un point essentiel pour la carte à micro-processeur : sa capacité de transporter une information secrète qui n'est jamais révélée (le porteur de la carte dispose d'un secret qu'il ne connaît pas).

En outre, lorsqu'on veut utiliser la fonction algorithmique avec le jeu secret de la carte, la fonction ne peut être appliquée qu'à des données de la carte, ce qui empêche la production éventuelle d'un certificat faux (parvenir à prouver la présence d'une donnée qui n'est pas en mémoire). Dans le même ordre d'idée, il est important que la fonction se base sur l'adresse de la donnée à traiter, ce qui empêche de pouvoir produire un résultat correct sur base d'une donnée qui est bien en mémoire mais pas à l'adresse désirée. Cet ordre ne souffre donc d'aucune lacune, cela en considérant que la fonction Télépass est cryptographiquement sûre.

4.2.3.3. Les zones du masque 4

Maintenant nous allons aborder les différentes zones du masque 4. C'est ainsi que l'on trouve :

- la zone secrète;
- la zone confidentielle;
- la zone d'accès;
- la zone transaction;
- la zone de lecture;
- la zone de fabrication.

Dans ces 6 zones, 2 zones sont principalement utilisées par le micro-processeur pour assurer le fonctionnement de la carte. Il s'agit de la zone de fabrication et de la zone d'accès. Les autres zones étant plus applicatives. En effet, les données qui y sont mémorisées sont propres à l'application. La figure 4.3 présente toutes les combinaisons d'accès possibles en se basant sur quatre possibilités : droits d'accès en écriture pour le micro processeur, pour le monde extérieur et de même pour les droits d'accès en lecture.

Droits d'accès				
du micro-processeur		du monde extérieur		
Lecture	Ecriture	Lecture	Ecriture	
NON	NON	NON	NON	1
NON	NON	NON	OUI	2
NON	NON	OUI	NON	3
NON	NON	OUI	OUI	4
NON	OUI	NON	NON	5
NON	OUI	NON	OUI	6
NON	OUI	OUI	NON	7
NON	OUI	OUI	OUI	8
OUI	NON	NON	NON	9
OUI	NON	NON	OUI	10
OUI	NON	OUI	NON	11
OUI	NON	OUI	OUI	12
OUI	OUI	NON	NON	13
OUI	OUI	NON	OUI	14
OUI	OUI	OUI	NON	15
OUI	OUI	OUI	OUI	16

Figure 4.3. : Droits d'accès.

Nous avons repris dans cette figure toutes les combinaisons possibles des droits d'accès. De ce fait, on trouve des combinaisons insensées. Par exemple, une zone qui ne peut être accédée en lecture, ni par le micro-processeur ni par le monde extérieur, n'a aucune utilité (écrire des informations qui ne peuvent jamais être lues n'a pas de sens). Il s'agit des lignes 1, 2, 5 et 6.

Nous devons, maintenant, éclaircir la sémantique du tableau. Quand on parle de lecture ou d'écriture du micro-processeur, il s'agit de lecture ou d'écriture effectuée de sa propre initiative, ceci excluant les accès qu'il fait suite à une initiative du monde extérieur. On peut considérer que lorsque le micro-processeur écrit des informations dans la mémoire, c'est dans le but de mémoriser des informations pour son fonctionnement. Il est indispensable qu'il puisse les lire ; les lignes 7 et 8 sont donc aussi éliminées.

Il reste ainsi dix possibilités. On peut cependant noter que certaines différences ne sont pas utiles. En effet, une zone qui respecte les droits d'accès des lignes 3 et 4 peut, sans problème pour la sécurité, appliquer les règles 11 et 12 où on a simplement donné le droit d'accès en lecture au micro-processeur. Les lignes 9 à 16 décrivent des droits d'accès tout à fait plausibles pour l'exploitation d'une zone :

9 une zone secrète totalement écrite à la personnalisation;
10 une zone secrète qui peut être complétée durant la vie de la carte (caractéristiques de la zone secrète du masque 4);
11 une zone d'accès libre en lecture totalement écrite à la personnalisation;
12 une zone réservée à l'application;
13 une zone secrète qui peut être complétée uniquement par le micro-processeur (génération d'une clé automatique);
14 une zone secrète qui peut être complétée durant sa vie aussi bien par le micro-processeur que par le monde extérieur;
15 une zone d'accès libre en lecture qui ne peut pas être complétée par le monde extérieur;
16 une zone totalement libre (la zone transaction du masque 4).

Plusieurs remarques doivent maintenant être faites. Premièrement, il est très important de souligner que le principe de non modifiabilité d'un mot écrit et validé est toujours présent. C'est essentiel, cela est à la base du principe de

preuve que peut fournir la carte. Par exemple : preuve d'une transaction monétaire, si le mot décrivant cette transaction pouvait être modifié, le montant de la transaction pourrait être changé !

Deuxièmement, nous avons montré que certaines combinaisons de droits d'accès peuvent être intéressantes alors qu'on n'en dispose pas dans le masque 4.

Troisièmement, nous n'avons pas abordé le problème de l'accès par clé. Il est parfois intéressant de protéger l'accès d'une zone, quand c'est autorisé, par une clé. Ceci était possible pour la zone transaction du masque 4 grâce à des indicateurs qui permettaient, à la personnalisation, de spécifier une protection par une clé. Ceci étant positif, il est regrettable que cette protection porte automatiquement sur toute la zone transaction. En effet, il serait peut-être souhaitable pour certaines applications d'avoir des zones de même type, au sens des règles d'accès énoncées par la figure 4.3 mais protégées par des clés de manière différente.

4.3. Le terminal

4.3.1. Introduction

Comme on le remarque sur la figure 4.4., le terminal est le second élément composant le réseau. Nous allons voir quelles sont ses fonctions ainsi que ce qu'implique sa position particulière.

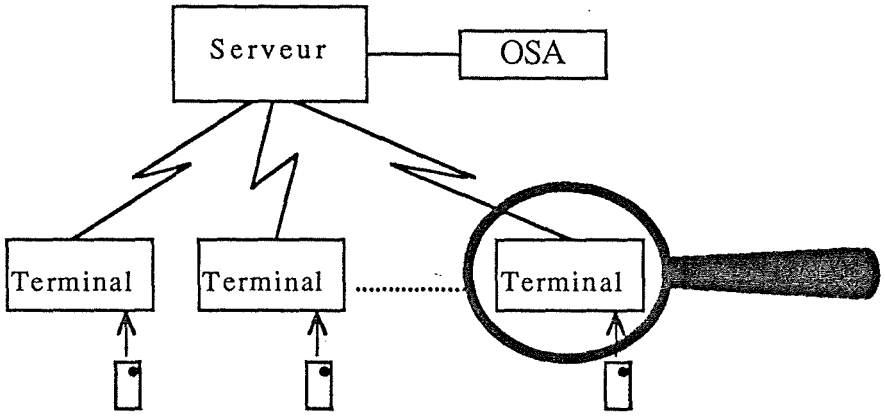


Figure 4.4. : Le terminal.

4.3.2. Les fonctions du terminal

Par sa position dans le réseau, le terminal occupe une place privilégiée. C'est lui qui est chargé de communiquer toutes les instructions à la carte, de gérer le dialogue avec l'utilisateur (un terminal comportant souvent un clavier/écran), de chiffrer et de déchiffrer les données pour les protéger durant leur "voyage" sur la ligne.

Sa place et ses fonctions permettent au terminal de connaître toutes les informations communiquées à la carte comme le code porteur, la clé secrète de l'émetteur, les données confidentielles qui sont lues ou écrites dans la mémoire de la carte. Le terminal peut donc constituer un point fragile du système. Nous étudierons diverses solutions à son sujet.

4.4. La ligne

La ligne est un élément passif du réseau. Les données qui y transitent sont protégées par un algorithme de chiffrement qui est le GOC. De plus le principe de signature vu au premier chapitre est aussi utilisé. En combinant ces deux systèmes, il est possible de lutter contre l'espionnage sur la ligne (comprendre le message), contre la perturbation (modification du message), contre l'insertion d'un message

4.5. Le serveur

La sécurité du serveur est plus difficile à étudier. De plus, on peut considérer qu'il s'agit d'autres problèmes de sécurité. La carte a été conçue pour être un support d'informations infraudable qui permet d'authentifier des personnes et de garantir l'écriture d'informations (certification). La carte est un élément de la sécurité du serveur mais elle n'est pas conçue pour solutionner tous les problèmes. C'est ainsi que l'on peut imaginer des attaques internes directement sur les données sans passer par les systèmes d'authentification mis en place. Il est donc très important de combiner à la carte d'autres techniques, dans le but de protéger le serveur.

4.6. L'OSA

4.6.1. Introduction

L'OSA occupe une place importante dans le réseau (figure 4.5). Sa tâche est essentielle. Nous allons, dans ce paragraphe, étudier les instructions qu'offre le OSA, plus spécialement celles qui concernent directement des fonctions de sécurité.

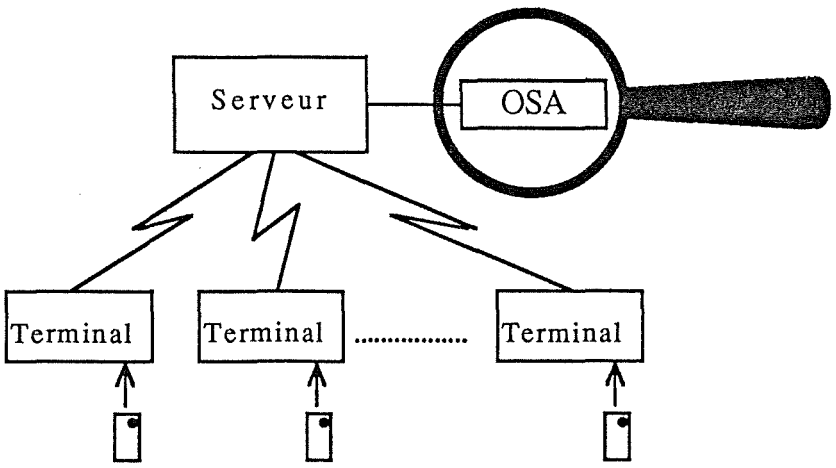


Figure 4.5. : L'OSA.

4.6.2. Les instructions de l'OSA

L'ordre de restauration et les ordres d'initialisation ne posent aucun problème de sécurité; en effet, ils ne divulguent aucune information confidentielle.

Nous pouvons ensuite aborder l'ordre de chiffrement et déchiffrement ainsi que l'ordre de demande de clé de chiffrement sur données internes ou externes. Ceci nous mène vers un premier point discutable dans l'OSA : d'un côté, nous avons des informations suffisamment importantes vu qu'elles ont été chiffrées; d'un autre côté, il n'y a aucune contrainte ou sécurité pour le déchiffrement de telles données. Ceci n'est pas en soi un problème mais cela signifie que l'on accorde toute confiance au serveur.

Une deuxième constatation est que l'on peut recevoir la clé de chiffrement sans aucune contrainte. Deux ordres sont disponibles pour obtenir une clé.

Le premier ordre permet de connaître la clé de chiffrement de la session (sur données internes). Il est indispensable lorsqu'on utilise pas le GOC fourni par l'OSA mais que l'on désire pouvoir disposer du système de génération de clé de chiffrement. Cet ordre n'est pas en soi un problème pour la sécurité.

Le second ordre (demande de clé de chiffrement sur données externes) permet d'obtenir la clé de chiffrement de n'importe quelle carte à partir du moment où l'on connaît ses paramètres de diversification et le nombre aléatoire utilisé. Cet ordre a été principalement créé pour le traitement d'informations mémorisées dans le serveur dans le cadre de messagerie électronique. Ceci est étonnant. En effet, pourquoi conserver un message chiffré si on ne soupçonne pas le serveur ? C'est donc en contradiction avec ce qui précède. Cet ordre est peut-être superflu dans la mesure où il est possible de déchiffrer le message dès sa réception et lorsque le message sera lu par le propriétaire de la boîte, il sera chiffré selon la clé de chiffrement de sa session.

Une autre série d'ordres est celle du contrôle de certificats. Elle permet de contrôler tous les certificats, ce qui, du point de vue de la sécurité, n'est pas gênant : il s'agit toujours d'un contrôle d'un certificat, et non une production d'un certificat qui permet une comparaison ultérieure. Ceci est très important car il n'est pas possible de produire un certificat avec un OSA dans des temps de réponse raisonnables. Une réserve peut toutefois être exprimée quant aux possibilités de déchiffrement au même titre que celles dites précédemment.

Le dernier ordre concerné par des problèmes de sécurité est celui de demande de clé émetteur. Cet ordre suppose à nouveau une totale confiance dans le serveur. Il est à remarquer que la protection faite par l'OSA en chiffrant automatiquement la clé n'est pas d'une grande utilité. Le GOC (algorithme de chiffrement utilisé) est connu et la clé utilisée peut l'être (décrit précédemment) !

4.7. Le module de contrôle et de sécurité

Le dernier composant qui a des fonctions importantes au niveau de la sécurité dans le réseau est, comme on peut l'observer sur la figure 4.6., le module de contrôle et de sécurité ou encore MCS.

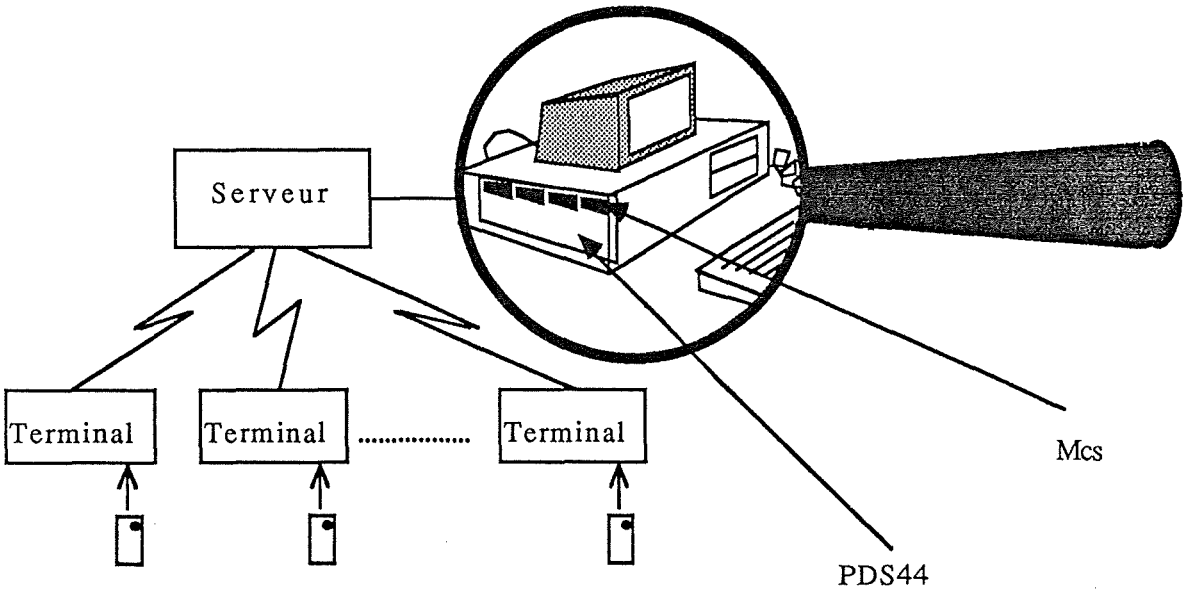


Figure 4.6. : Le module de sécurité.

Que peut-on dire du MCS ? En premier lieu, on doit souligner sa conception hardware. En effet au niveau hardware, le principe de la carte a été conservé, un micro-processeur qui n'autorise qu'un nombre limité d'opérations d'accès à une mémoire permanente. Il n'y a cependant aucune authentification de la personne qui tente d'accéder à sa mémoire.

Deuxièmement, nous trouvons sur le MCS une structure d'informations relativement précise où des données destinées au micro-processeur permettent à ce dernier d'assurer une sécurité maximale pour le secret de base d'une application (secret Mère). C'est ainsi qu'à aucun moment il n'est possible de connaître le jeu secret Mère. De plus on peut très bien choisir ce qu'un OSA peut faire ; par exemple, autoriser le contrôle de certificat ne permet en aucun cas de produire une clé de chiffrement.

Troisièmement, on constate qu'un MCS permet de connaître toutes les clés émetteurs du parc de cartes qu'il gère, à partir du moment où il contient un bloc conçu dans ce but. Ceci n'est pas un problème en soi puisque c'est son objectif. Il est donc très important que l'OSA et ces MCS soient dans un local plus que hautement surveillé !

Dans un quatrième point, nous soulignerons un problème d'utilisation actuel des MCS qui augmente le risque de perte ou vol présenté dans le paragraphe précédent. Les blocs secrets sont toujours écrits lors de séance de personnalisation du MCS chez le fabricant. Ceci peut être gênant au niveau des transports ultérieurs du composant.

Un dernier point que nous mentionnerons est celui de production de clé émetteur. A ce sujet, il peut être étonnant que la clé soit ainsi délivrée, sans chiffrement.

4.8. Résumé et remarques

Ci-dessous, nous allons reprendre chaque remarque ou réserve que nous avions exprimées précédemment. Ensuite un commentaire sera développé.

- schéma hardware : impossible de connaître des informations par voie physique directe;
- faille possible de l'implémentation du masque de la carte;
- pas de chiffrement des données lors de la lecture;
- pas de chiffrement des données lors de l'écriture;
- pas de chiffrement des clés lors de leur présentation;
- utilité de l'ordre de validation de clé en lecture;
- nécessité de l'ordre de validation en écriture;
- efficacité cryptographique de la fonction Télépass;
- absence de certains types de zones;
- les paramètres de protection portent automatiquement sur toute la zone transaction;
- le terminal est chargé des opérations de chiffrement et déchiffrement;
- le terminal dispose de toutes les informations confidentielles communiquées à la carte;
- efficacité cryptographique du GOC;
- problème de sécurité générale du serveur;
- le serveur peut, grâce à l'OSA, déchiffrer tout message;
- la clé de chiffrement peut être livrée sans aucune contrainte;
- les clés émetteurs sont délivrées sans aucun contrôle;
- aucune authentification lors d'utilisation d'un MCS;
- danger de perte des MCS;
- le transport des secrets est toujours physique, pas de télé-écriture dans les MCS;
- pas de chiffrement de certains résultats de la fonction algorithmique (lors de calcul de clé émetteur par exemple).

La liste établie ci-dessus permet avec un peu d'imagination de percevoir différents moyens d'attaques possibles sur la carte. On pourrait penser, vu la longueur de cette liste, que nous condamnons la carte à micro processeur. Ce serait une erreur. En effet, les critiques ici présentées reposent très souvent sur base d'attaques internes. Si nous émettons des réserves sur des données confidentielles qui ne sont pas chiffrées, c'est qu'en cas de "piratage" d'un élément interne du réseau, on peut connaître des informations essentielles pour le traitement d'une carte (ex. enregistrement de tous les ordres qu'un terminal envoie à la carte).

Nous pouvons encore rajouter quelques considérations annexes au système de sécurité :

- regarder le porteur de la carte taper son code personnel et voler sa carte;
- chantage sur le personnel responsable de la gestion du réseau et du parc de cartes;

- identification correcte du porteur de la carte lors de la distribution de la carte.

Ci-dessous, la figure 4.7. reprend, de manière imagée, les différentes attaques que nous avons relevées en fonction des caractéristiques de la carte à micro-processeur.

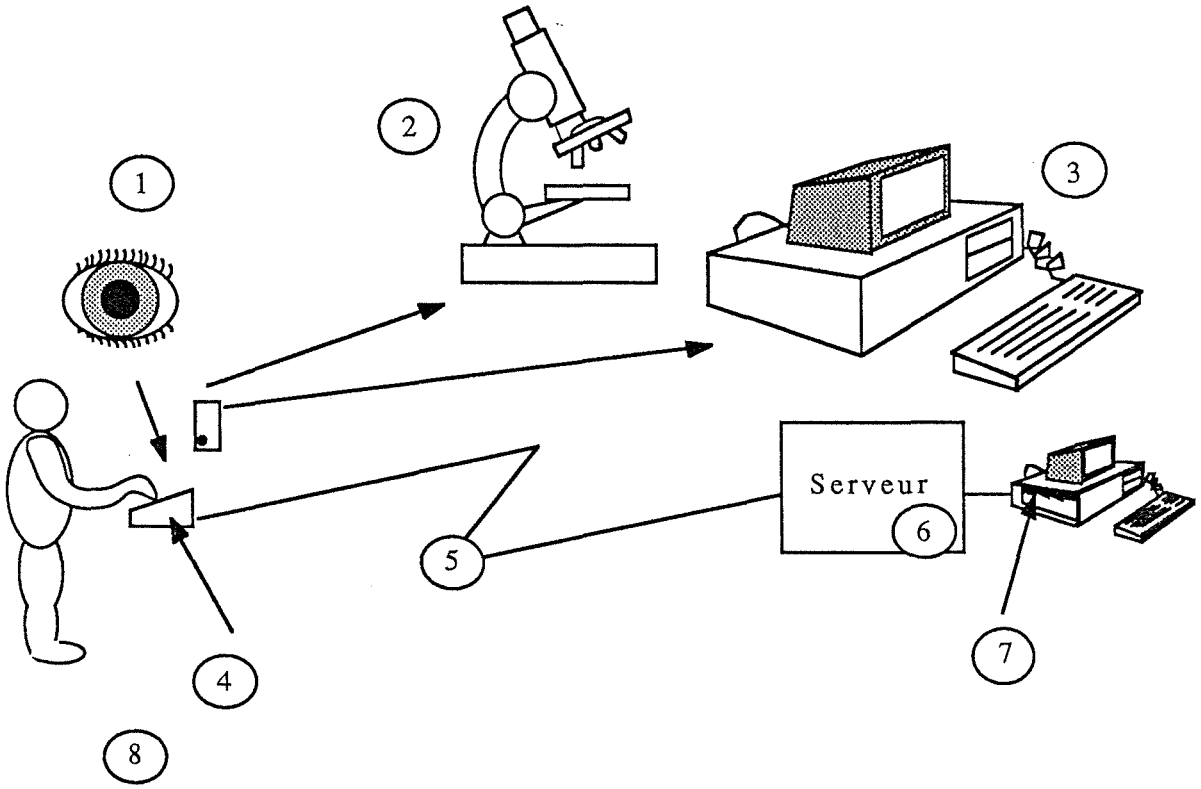


Figure 4.7. : Les attaques du système de sécurisation à base de cartes à micro-processeur.

1 observation du code porteur et vol de la carte;
2 attaque hardware de la carte;
3 attaque software de la carte (faute dans l'implémentation du masque)
4 piratage du terminal;
5 cassure du système cryptographique GOC;
6 intrusion dans le serveur;
7 vol d'un MCS;
8 identification du porteur de la carte lors de la remise de celle-ci.

Chapitre 5

Différentes améliorations de la solution OSA et de la carte

5.1. Introduction

Dans ce chapitre nous allons présenter différentes suggestions dans le but d'améliorer la sécurité offerte par la solution OSA. Nous allons nous efforcer d'englober dans la carte et le Mcs certains mécanismes qui s'imposeront à l'utilisateur, ceci dans le but d'empêcher le maximum de "tricheries" et cela notamment aux niveaux internes (serveur, terminaux,...).

Nous nous baserons sur un réseau plus compliqué. Ce réseau, présenté par la figure 5.1. possède un niveau supérieur. En effet, dans bon nombre d'applications susceptibles d'utiliser la carte, il y a plusieurs serveurs, qui desservent un parc de terminaux d'une région (cas bancaire par exemple). C'est ainsi qu'il y a la notion de serveur principal qui "chapeaute" les autres serveurs et qui est à la base de l'organisation du réseau. Il est chargé de la distribution des clés secrètes et d'autres informations telles que les listes noires (liste de cartes à invalider pour cause de vol par exemple).

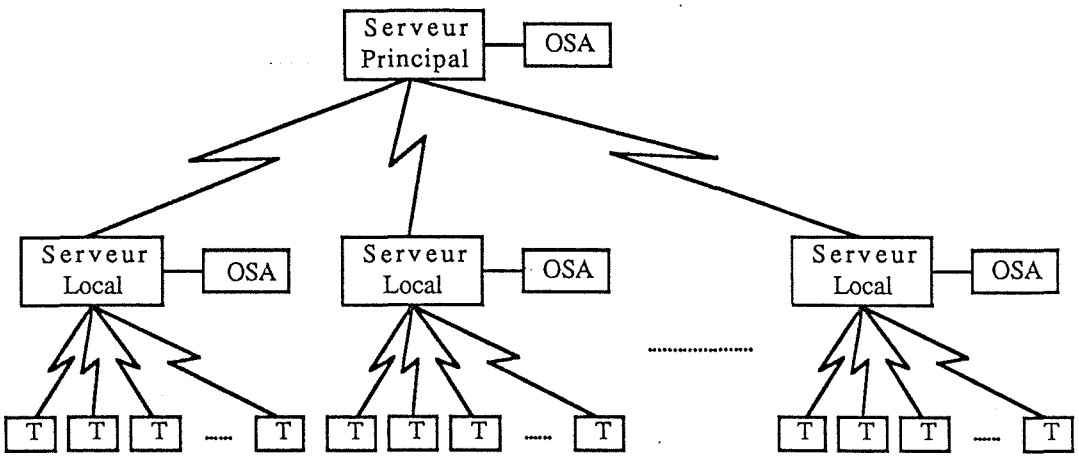


Figure 5.1. : Réseau avec un serveur principal.

Dans ce chapitre nous n'étudierons pas de parades aux attaques de type hardware et software (basée sur une éventuelle mauvaise implémentation du masque de la carte), de même que nous n'aborderons pas le problème de l'identification du porteur lors de la remise de la carte, tout ceci dépassant le cadre de ce mémoire.

5.2. L'authentification du porteur

L'authentification du porteur de la carte est un problème délicat au niveau de la sécurité. Actuellement le procédé le plus répandu est le code secret. Il est généralement composé de quatre chiffres.

Ce système de code secret est en fait un compromis entre le niveau de sécurité et la facilité d'implémentation et d'utilisation. Il présente néanmoins quelques lacunes.

En premier lieu on peut remarquer que le code secret peut être utilisé par une tierce personne. Ceci peut être grave dans le cas où il s'agit d'une personne qui a "volé" le secret et la carte, ou lorsque le porteur donne son secret à une autre personne qui dispose ainsi d'un droit d'utilisation d'une carte alors qu'elle n'y est, à priori, pas autorisée. Dans ce dernier cas c'est le "prestataire" de service qui se trouve éventuellement lésé.

En second lieu, ce système peut ne pas être pratique en fonction de l'utilisation de la carte. En effet, cette utilisation peut être occasionnelle et de ce fait il est difficile de mémoriser le secret surtout lorsqu'on dispose de plusieurs cartes dont les secrets sont tous différents. Ce dernier point pourrait être un frein considérable au développement de l'utilisation des cartes à micro-processeur.

Ci-dessous nous résumons les problèmes posés par l'utilisation actuelle de code secret :

- vol et utilisation du secret par piratage du terminal;
- vol et utilisation du secret par observation de la frappe sur le clavier;
- don de son secret à une personne qui n'a pas les droits d'utilisation requis;
- difficulté de mémorisation du secret;
- vol et utilisation du secret car ce dernier est noté par l'utilisateur sur un document proche de la carte.

5.2.1. Le vol du secret par piratage du terminal

Pour contrer le vol du secret par piratage du terminal on peut envisager plusieurs solutions. Une première étant de ne pas passer par le terminal pour donner le code secret à la carte. Il s'agit d'équiper la carte d'un mini clavier, ce qui permet d'introduire le code. La figure 5.2. présente cette première solution où seule une partie de la carte est introduite dans le lecteur, le "clavier" restant accessible à l'utilisateur.

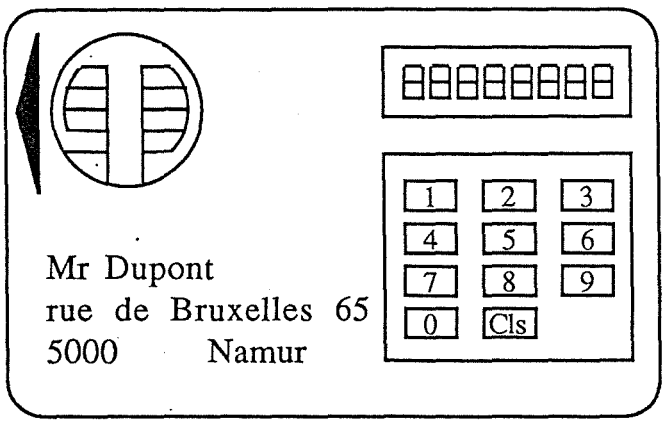


Figure 5.2. : La carte à micro-processeur équipée d'un mini-clavier.

Une seconde solution est d'avoir un système où le code secret est différent à chaque présentation. De ce fait, le vol du secret n'est d'aucune utilité car non réutilisable. Cette solution bien qu'alléchante est assez difficile à réaliser. En effet, un tel système se base sur un protocole qui permet de modifier la clé à chaque présentation. Ce protocole devra être "exécuté" par le porteur lui-même. De plus, pour être valable au niveau de la sécurité, il ne doit pas être trop simple. Ceci complique donc l'utilisation de la carte et augmente grandement les problèmes de mémorisation du secret.

Une troisième solution est d'établir un système qui permet au terminal de prouver à la carte qu'il est "honnête". Ceci passe par un système d'authentification. On peut imaginer l'insertion d'un Mcs dans le terminal. La carte, de son, coté ne fonctionne pas tant que le message d'authentification n'est pas correct. Bien que cette solution est de toute apparence intéressante, elle n'en n'est pas moins inefficace et dans certains cas inapplicable. Pourquoi ?

Ce qu'il faut se rappeler, c'est que nous nous basons sur une attaque par piratage du terminal. On peut en imaginer de deux types : la première étant d'installer un faux terminal (il n'est même pas connecté au serveur), la seconde se basant sur l'introduction dans un "vrai" terminal d'une petite routine mémorisant le code secret.

Au niveau du premier type il est évident, dans la mesure ou le système d'authentification du terminal est bien élaboré, que ce terminal ne sera pas capable de s'authentifier. Mais qu'est-ce qui empêche ce terminal de demander le code porteur sans que la carte fonctionne ?

C'est la même chose au niveau du second type de piratage. En effet qu'est-ce qui empêche l'introduction d'une routine pirate dans un terminal qui dispose d'un système d'authentification ?

L'inapplicabilité ? Dans certaines applications, il peut être très intéressant d'autoriser des opérations avec la carte à partir du domicile du porteur, par exemple, grâce à un Minitel. Il est évidemment peu raisonnable d'imaginer une authentification du terminal dans ce cas ; il serait nécessaire d'équiper tous les

terminaux de Mcs mémorisant les secrets de base de toutes les applications utilisées par le propriétaire du Minitel !

5.2.2. Le vol du secret par observation de sa frappe et les problèmes de mémorisation

Le deuxième problème mentionné est le vol du secret par observation de la frappe de ce dernier. Il annule quelque peu la première solution évoquée (celle du clavier sur la carte). Par contre la seconde solution (code secret différent à chaque fois) est quant à elle toujours valable. Mais, lorsqu'on ajoute le problème suivant de divulgation volontaire et les problèmes de mémorisation (le fait de noter le code trahit la difficulté de mémorisation), on se rend compte que cette solution n'est plus valable. En effet un tel protocole peut être divulgué à une autre personne et pose encore plus le problème de mémorisation.

5.2.3. La solution idéale ?

La solution idéale consiste à se baser sur des codes biométriques tels que les empreintes digitales. Seulement, en fonction d'un piratage éventuel d'un terminal, cette fonction doit être réalisée par la carte ! Nous imaginerons donc une carte incorporant un dispositif de reconnaissance d'empreintes.

Ceci clôture la présentation de solutions pour le code porteur. On peut remarquer qu'aucune n'est théoriquement satisfaisante si ce n'est la dernière. Malheureusement il est clair qu'actuellement elle se heurte à un problème technologique.

Ce que nous voudrions rappeler c'est le principe même de sécurité. Nous devons créer un système tel que les coûts d'un attaquant quelconque soient plus importants que les bénéfices qu'il peut retirer. De plus, l'investissement au niveau de la sécurité doit être inférieur au coût du dommage qui sera contré. C'est ainsi que le degré de sécurité d'une application dépendra de ses caractéristiques propres. Un code secret peut, dans certains cas, être amplement suffisant.

5.3. La carte, diversification et chiffrement

5.3.1. Introduction

Dans le chapitre précédent nous avions relevé certaines lacunes quant à la gestion des secrets et des différents chiffrements. Nous allons vous présenter une nouvelle solution apportant plus de sécurité aux différentes données.

5.3.2. Une nouvelle fonction cryptographique

Dans un premier temps nous allons définir une nouvelle fonction cryptographique. Bien entendu nous ne définissons que les caractéritiques "extérieures" de cette fonction.

Tout d'abord il est important de pouvoir diposer d'une fonction cryptographique publique à clé secrète, la carte étant tout à fait appropriée à ce type de fonction. De plus il est souhaitable que la clé de chiffrement et celle de déchiffrement soient identiques. Si ce n'était pas le cas, il faudrait que l'on puisse disposer d'une fonction qui peut aisément déterminer la clé de déchiffrement à partir de la clé de chiffrement.

La fonction est composée de deux sous-fonctions. Il y a la fonction de chiffrement/déchiffrement proprement dite, notée F sur la figure 5.3 ainsi qu'une fonction, notée G, qui modifie le jeu secret selon des paramètres tous facultatifs : un nombre aléatoire, une adresse et deux indicateurs qui provoquent une transformation.du jeu secret.

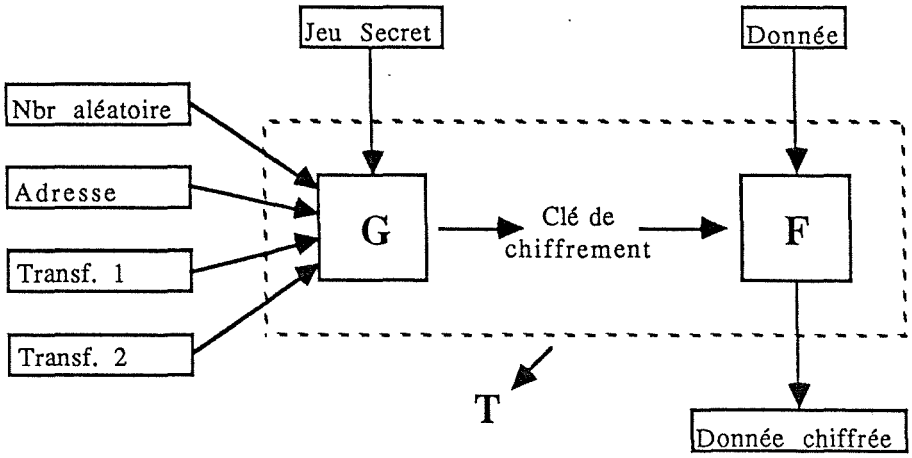


Figure 5.3. : Structure d'une nouvelle fonction cryptographique.

Nous expliquerons son utilisation à travers les paragraphes suivants ainsi que nous justifierons sa construction. Nous noterons l'ensemble de la fonction T qui est à 6 arguments (nombre aléatoire, adresse, Tr 1, Tr 2, jeu secret, donnée), les quatres premiers étant facultatifs.

5.3.3. La clé émetteur

Nous avions déploré au niveau de la carte masque 4 que la clé émetteur était la même à chaque présentation. Ceci constitue une faiblesse qui n'existe pas, par exemple, dans la technique de certification qui, elle, fait intervenir un nombre aléatoire.

Le but recherché est que seuls le Mcs qui produit la clé émetteur et la carte à laquelle elle est destinée puissent connaître cette clé.

La procédure est décrite par la figure 5.4. Dans un premier temps (1) la clé émetteur est calculée en fonction de paramètres de diversification dans le Mcs. Le résultat de la diversification ne peut ni être lu, ni être comparé. Dans un second temps (2) le Mcs calcule le jeu secret de la carte sur base des mêmes paramètres de diversification . Dans un troisième temps (3) la clé émetteur est chiffrée à partir du jeu secret, d'un nombre aléatoire et en provoquant la transformation 1. La carte, de son coté, dispose aussi du jeu secret et du nombre aléatoire. Il est donc possible pour elle de déchiffrer la clé émetteur.

Cette opération de déchiffrement s'effectue automatiquement lors de la présentation de la clé émetteur.

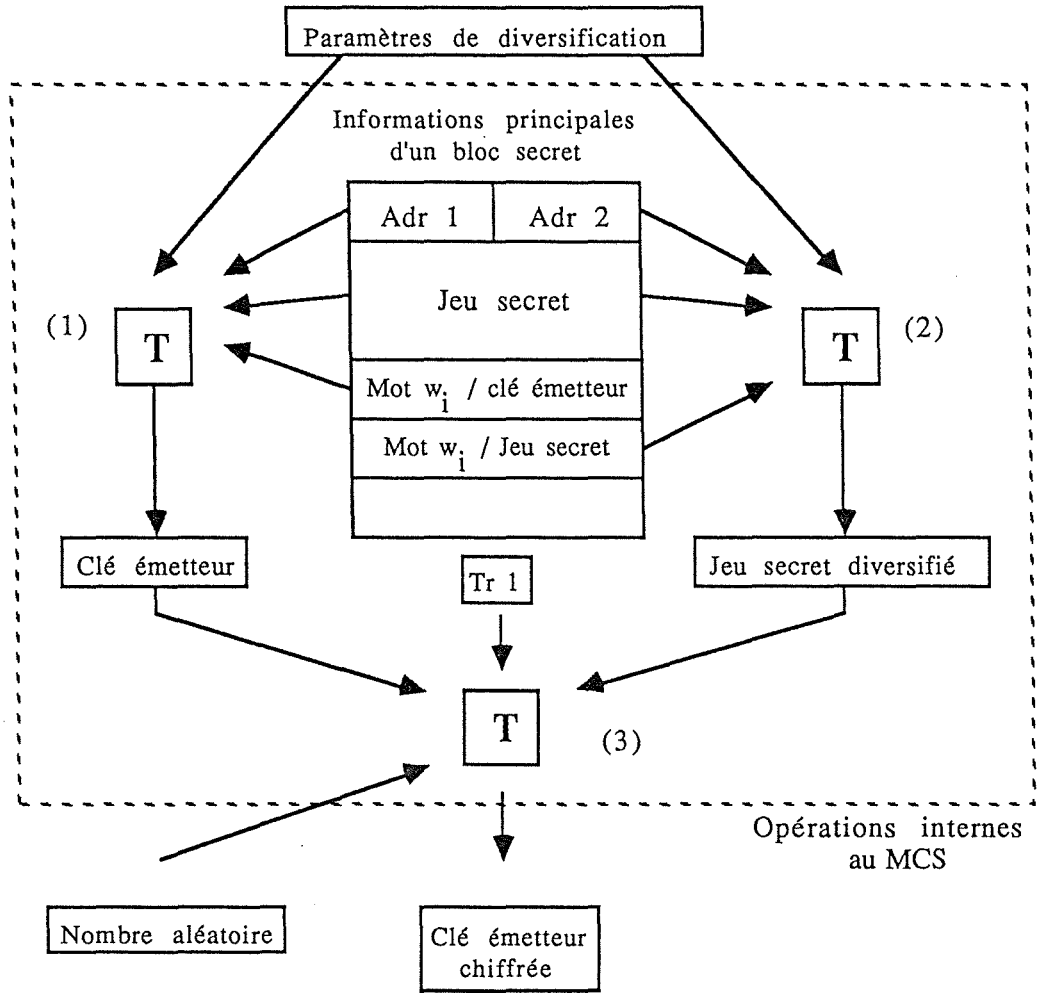


Figure 5.4. : Processus d'émission d'une clé émetteur chiffrée.

5.3.4. Chiffrement systématique des données d'une zone protégée par une clé

En ce qui concerne le problème de chiffrement de données mémorisées dans la carte, il faut remarquer une chose : si ces données ne sont pas dans une zone secrète (accesible seulement par le micro-processeur) c'est qu'elles seront lues au moins une fois par le monde extérieur et cela dans le but d'être traitées. Deux possibilités se présentent : elles seront traitées, soit par le serveur, soit par le terminal. Dans ce dernier cas le chiffrement n'a pas d'utilité.

Dans le cas où le traitement est effectué par le serveur, on peut éventuellement considérer qu'il est intéressant que les données soient chiffrées, ceci dépendant de l'application dans le cas où l'on estime devoir se protéger du terminal. Il sera donc souhaitable de ne pas chiffrer systématiquement toutes les données appartenant à une zone qui est protégée par une clé. On peut imaginer un nouvel attribut d'une zone transaction : chiffrement des données. Il est bien entendu que le chiffrement se fait avec l'utilisation d'un nombre aléatoire (seul paramètre de la fonction G).

Nous devons concevoir un bloc pour le chiffrement/déchiffrement des données que l'on veut écrire ou lire dans la carte. On peut déjà percevoir l'utilité de la transformation 1 appliquée par la fonction G au jeu secret lors du chiffrement de la clé émetteur. Cette application a pour seul but d'empêcher le déchiffrement de la clé émetteur par l'utilisation d'un bloc prévu pour le déchiffrement des données d'une zone protégée.

5.3.5. La certification

Reste bien entendu la certification qui est essentielle dans le principe de la carte. Grâce à ce système, il est possible de réaliser des fonctions comme l'authentification, la génération de clé de chiffrement ainsi que le principe de signature électronique. La certification pose toutefois un léger problème. En effet, à partir du moment où la fonction cryptographique permet de chiffrer les données, il n'y a plus moyen de distinguer cette dernière opération de celle de certification. Le certificat n'est autre que le chiffrement de données.

C'est pour cette raison que nous ne faisons pas intervenir l'adresse lors du chiffrement simple des informations, ceci n'étant pas nécessaire pour le chiffrement alors que c'est essentiel pour la certification. Par conséquent il sera bel et bien impossible de fournir un certificat via la fonction de chiffrement.

5.3.6. Chiffrement personnel et "courrier électronique"

5.3.6.1. Introduction

A présent nous diposons d'un sytème où l'on peut distinguer deux cas :

- Le terminal tout comme le serveur ne connaissent pas le contenu du message (cas de la clé émetteur);
- Seul le serveur peut connaître les informations (cas où la zone exige un chiffrement).

Il reste une autre possibilité non exploitée : seul le terminal peut connaître les informations. En effet, on peut imaginer un système de courrier éléctronique où le contenu des messages ne peut pas être connu par une personne autre que les deux correspondants qui diposent bien entendu d'une carte à micro-processeur. On peut aussi concevoir un cas où le porteur d'une carte à micro-proceseur veut mémoriser dans le serveur des informations confidentielles qu'il veut être le seul à pouvoir les déchiffrer.

5.3.6.2. Le chiffrement "courrier électronique"

La taille des informations dans le cadre du courrier électronique est variable. Par conséquent le chiffrement de telles données peut être assez long. Il n'est donc pas envisageable de réaliser le chiffrement via la carte. Nous allons établir un système qui génère une clé de chiffrement qui peut être communiquée de manière univoque à un et un seul destinatire.

La première partie de la procédure est présentée par la figure 5.5. dans une première étape, on établit une clé de chiffrement par tirage d'un nombre aléatoire (1). L'algorithme que sera utilisé pour le chiffrement est quelconque, si ce n'est qu'il est souhaitable que la clé de chiffrement et de déchiffrement soient identiques. Si ce n'est pas le cas, il est impératif de pouvoir disposer d'une fonction pour déduire la clé de déciffrement à partir de la clé de chiffrement.

Les paramètres de diversification de la carte **réceptrice** doivent être connus. De plus, ils l'identifient de manière sûre. Ces paramètres sont chiffrés

(2) de même que la clé de chiffrement choisie préalablement (3). Le message est donc acompagné des paramètres de diversification de la carte **émettrice** et le nombre aléatoire (en clair), ainsi que les paramètres de diversification de la carte **réceptrice** et la clé de chiffrement, tous deux chiffrés. Les deux transformations sont appliquées lors du chiffrement.

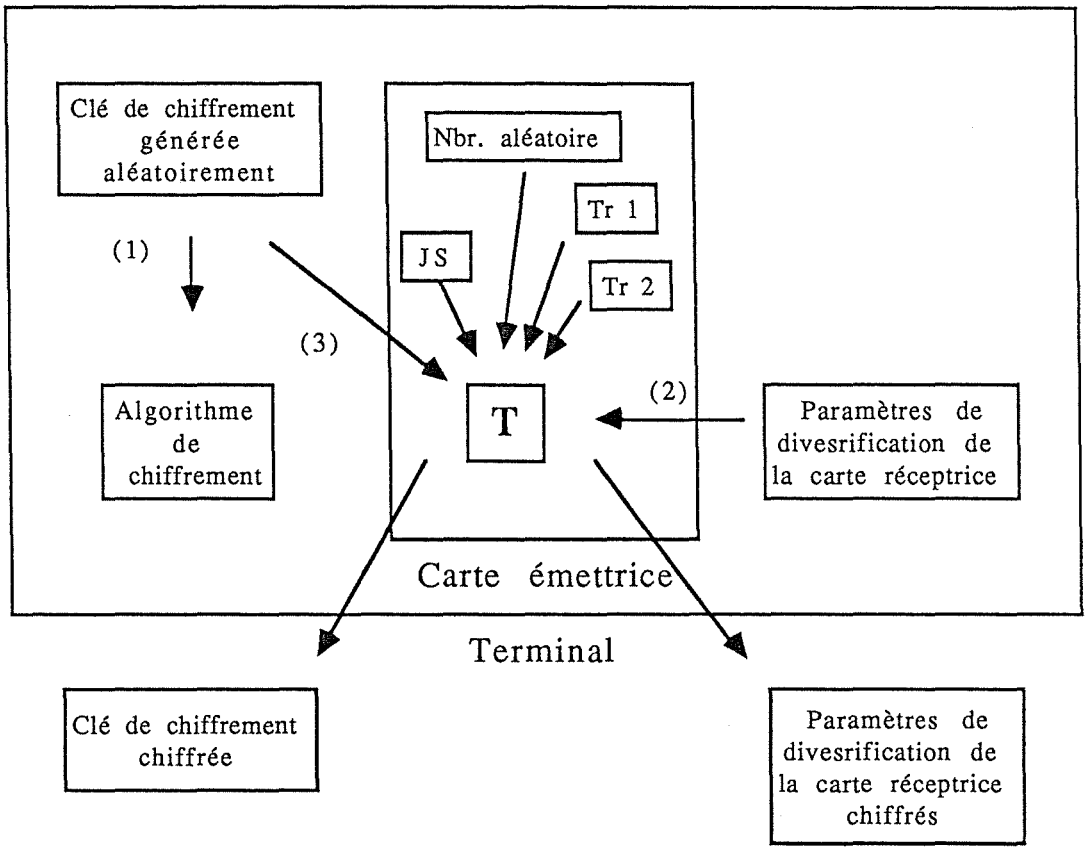


Figure 5.5. : Première partie de la procédure de génération de clé pour un système de courrier électronique.

Un bloc secret de "traduction" de la clé de chiffrement est conçu. Tout d'abord le jeu secret de la carte **émettrice** est calculé (1 figure 5.6). Grâce à ce dernier, il est possible de déchiffrer les paramètres de diversification de la carte **réceptrice** (2) et la clé de chiffrement (3). Avec ces paramètres de diversification ainsi déchiffrés, le jeu secret de la carte **réceptrice** est calculé (4). Il ne reste plus qu'à chiffrer la clé de chiffrement (5). Le résultat de ce chiffrement est le seul résultat de la fonction T lisible par le monde extérieur.

Ceci garantit que la clé de chiffrement ne sera connue que par les deux correspondants.

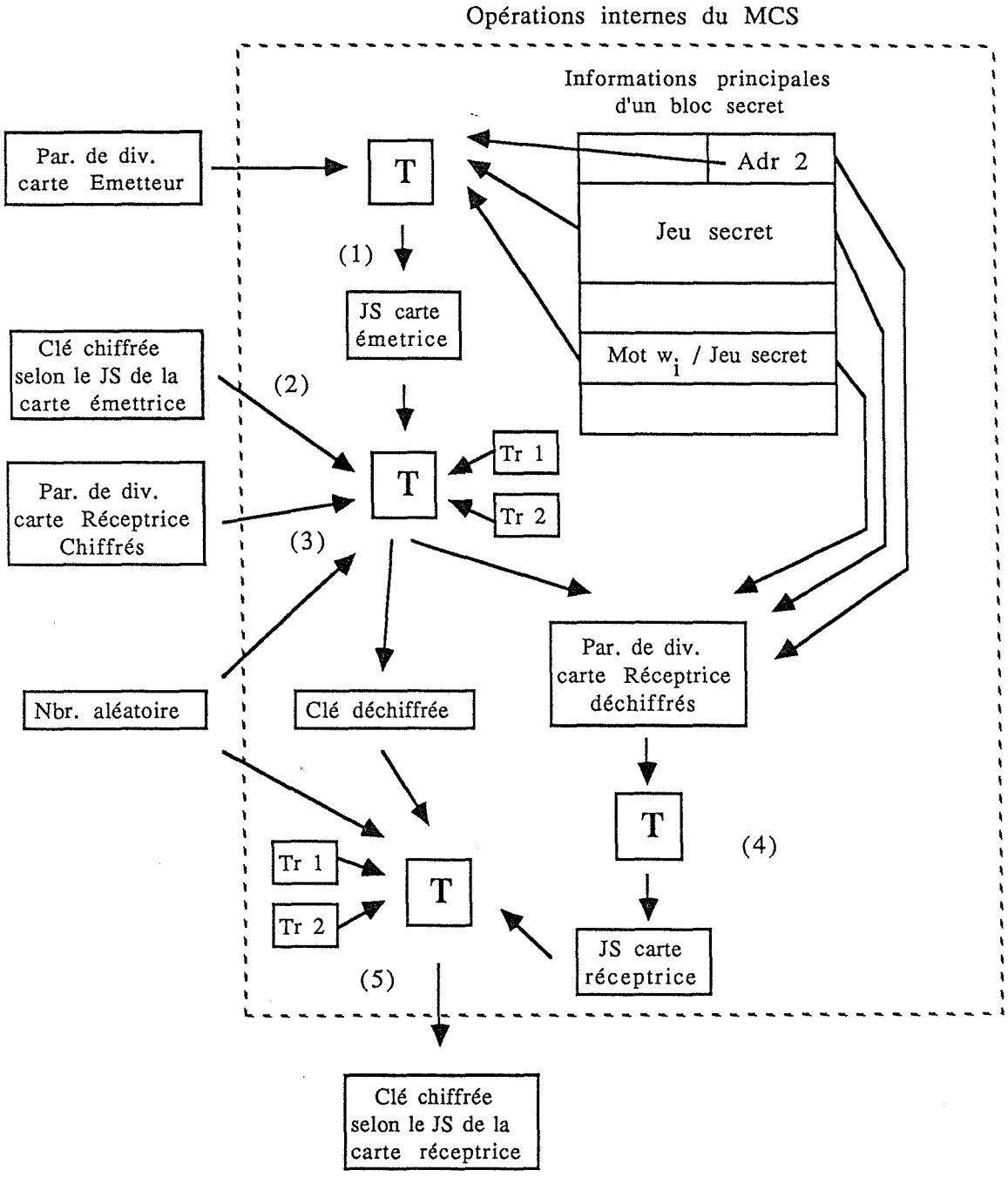


Figure 5.6. : Seconde partie de la procédure de génération de clé pour un système de courrier électronique.

La carte **réceptrice** dispose d'un ordre de déchiffrement. C'est pour cette raison que les deux transformations ont été appliquées sur la clé, dans le but d'empêcher la carte de déchiffrer une clé émetteur. Les paramètres de diversification de la carte réceptrice sont chiffrés pour éviter qu'ils ne soient modifiés par le serveur dans le but de déchiffrer le message à l'aide d'une carte "complice".

5.3.6.3. Le chiffrement de données personnelles

En ce qui concerne la production d'une clé de chiffrement réservée pour le porteur de la carte, plusieurs méthodes peuvent être utilisées. Une solution simple est de combiner un certificat classique, qui comprend donc un nombre aléatoire, avec un code qui serait demandé au porteur de la carte. Le nombre aléatoire est mémorisé en clair, avec le message, dans le serveur.

5.3.7. Conclusion sur la nouvelle fonction cryptographique

La nouvelle fonction cryptographique est invoquée à différentes reprises. Nous allons vous présenter les différents ordres qui s'en servent ainsi que les paramètres de la fonction G qu'ils communiquent.

Ordre	Nbr E	Adresse	Tr 1	Tr 2
Ecriture d'une nouvelle clé secrète (1)	X			X
Présentation de clé Emetteur	X		X	
Lect. / Ecrit. dans une zone avec chiffrement	X			
Ordre de calcul pour les certificats	X	X		
Ordre de chiffrement / déchiffrement	X		X	X

Nous pouvons constater dans cette énumération que les combinaisons des paramètres de la fonction G sont à chaque fois différents, ce qui garantit qu'une fonction ne peut pas simuler une autre.

5.4. Le modules de contrôle et de sécurité

5.4.1. Introduction

Nous n'avons pas encore tenu compte du fait de la modification apportée sur notre réseau type : la présence d'un serveur principal. C'est un premier point que nous abordons : comment distribuer des blocs secrets dans les différents serveurs ?

Il est évident que les nouvelles fonctions introduites dans les paragraphes précédents impliquent des changements au niveau de la définition des blocs. Nous développerons ces nouveaux blocs.

(1) Cette possibilité sera développée ultérieurement au paragraphe 5.4.3.

5.4.2. Le transfert de bloc secret

Le mécanisme est relativement simple. Nous allons concevoir les Mcs comme les cartes, à savoir qu'ils ont une zone de fabrication et une zone de secret. La zone secrète contient seulement le jeu secret ainsi qu'une adresse et un mot. Avec ce mécanisme on peut créer un réseau à plusieurs "niveaux". En effet (figure 5.7.) le jeu secret d'un niveau I est un jeu secret de base par rapport au niveau I+1 et est un jeu secret diversifié par rapport au jeu secret I-1 (qui est donc le jeu de base pour lui).

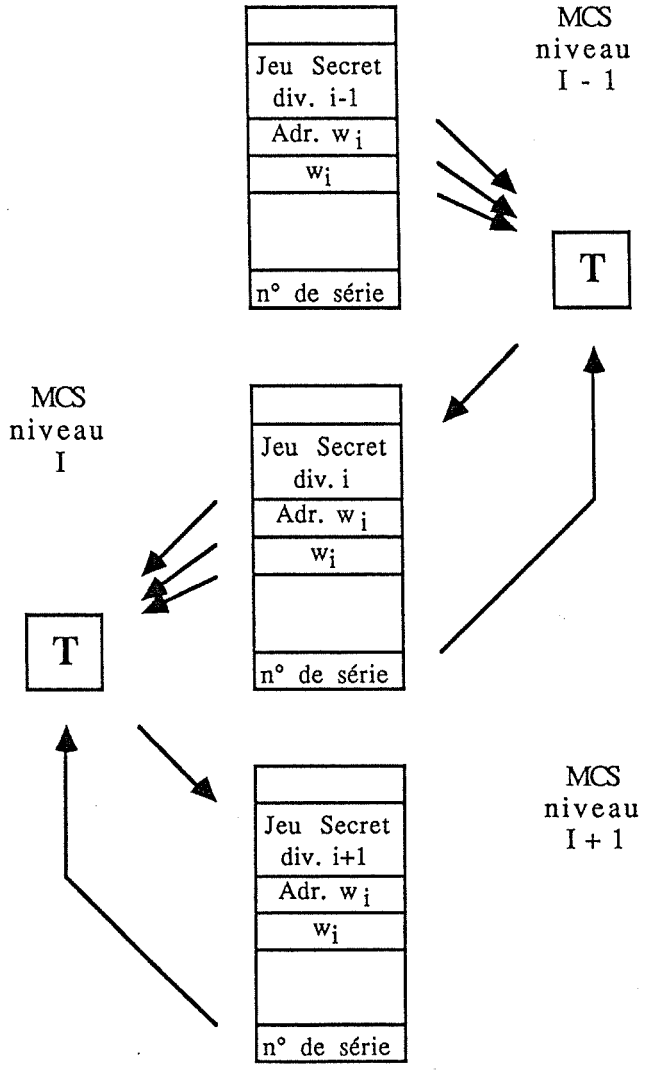


Figure 5.7. : système de distribution des jeux secrets qui protègent le transfert des blocs secrets.

Deux ordres seront créés : l'ordre d'émission d'un bloc secret et l'ordre d'introduction d'un bloc secret. Le premier consiste à chiffrer un bloc secret que le MCS contient, selon les paramètres de diversification du MCS destinataire, le second d'écrire dans la mémoire un bloc chiffré qu'on vient de recevoir. Cette écriture se charge automatiquement de la validation de chaque mot du bloc si la zone contrôle(1) confirme la bonne écriture du bloc.

Le Mcs du serveur principal sera le seul où l'écriture directe (en clair) sera autorisée. Il est peut-être intéressant de protéger l'accès de ce Mcs par des cartes à micro-processeur qui permettent d'authentifier les seules personnes qui sont autorisées à concevoir un code secret pour une application (en général des responsabes de la société utilisatrice).

5.4.3. La modification des secrets de la carte

Nous pouvons toutefois regretter que la solution présentée ci-dessous, porte seulement sur les MCS. En effet, pourquoi s'arrêter en "si bon chemin" en ne laissant pas la possibilité de changer des secrets au niveau de la carte ?

Le problème réside dans la correspondance du jeu secret mémorisé dans la carte et celui du MCS. En fait il n'y en a pas ! Par conséquent il faut imaginer une autre procédure.

Tout d'abord, nous devons souligner que les clés émetteur ainsi que le jeu secret sont issus du même jeu secret Mère (voir principe de diversification au paragraphe 1.8.8.). Il est donc indispensable de modifier les trois secrets en même temps (clé émetteur primaire et secondaire et le jeu secret). N'oublions pas aussi que la clé émetteur secondaire est facultative. En outre il n'y a pas de bloc qui permet de générer le jeu secret de la carte fille.

Nous devons remarquer que le code secret du porteur est un secret particulier. Il ne peut pas être changé comme les autres car il y a un problème pour sa communication au porteur. Ce n'est que par une initiative de ce dernier qu'il peut être changé. Deux possibilités se présentent selon qu'on a conçu une

(1) La présentation du bloc secret et de ses zones sera abordée au paragraphe 5.4.4.

carte avec un mini clavier ou non. S'il n'y a pas de clavier c'est le terminal qui saisit le code porteur en clair et qui l'introduit à la carte. S'il y a un mini clavier le terminal lance un ordre de modification du code porteur à la carte qui, elle, par son propre écran, invite le porteur à changer son code. Cette dernière méthode est évidemment plus sûre.

Nous introduisons quelques contraintes : le Mcs qui désire écrire de nouveaux secrets doit obligatoirement disposer de l'ancien et du nouveau bloc de certification, du nouveau bloc de clé émetteur primaire et secondaire si la carte dispose d'une clé secondaire. Ces contraintes sont réalistes car, à partir du moment où l'on juge un centre suffisamment sûr pour qu'il puisse disposer d'un bloc d'émission de clé émetteur, on peut certainement lui "confier" un bloc de certification.

A partir du bloc de certification, il est possible de connaître le jeu secret de la carte (n°1 sur la figure 5.8) et de générer le nouveau (2). Les nouveaux blocs de clé émetteur permettent de générer les nouvelles clés émetteur diversifiées (3). Elles seront chiffrées (4) ainsi que le Jeu secret. Il est à noter que l'entête du secret est insérée automatiquement(1). Le chiffrement se fait avec un nombre aléatoire, l'ancien jeu secret et la transformation 2. La carte écrit les nouveaux secrets dans la zone secrète. Si un des blocs manque à "l'appel", les blocs écrits ne seront pas validés et, par conséquent, ne seront par actifs.

(1) La mémorisation des secrets dans la zone secrète de la carte sera présentée au paragraphe 5.6.2.

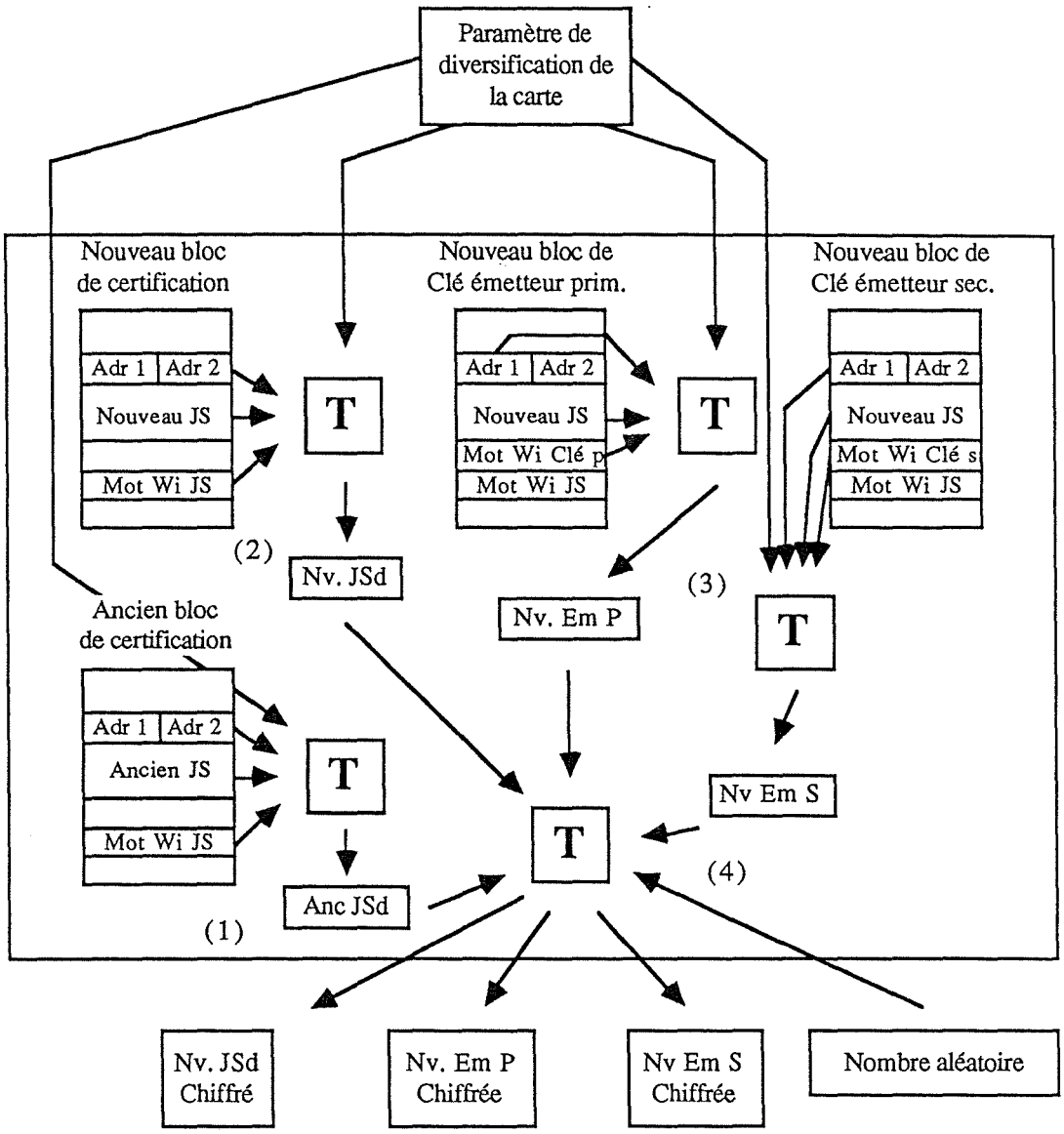


Figure 5.8. : Procédure de modification du jeu de clé d'une carte.

Un mécanisme est aussi fourni pour le cas où l'on veut ajouter une clé émetteur secondaire à une carte qui n'en dispose pas. Le mécanisme est du même type que le pécédent. Le MCS doit disposer d'un bloc de certificat du même TYPE(1) que ce lui de la carte ainsi que d'un bolc de clé émetteur secondaire. Le chiffrement s'opère de la même manière mais sur base du jeu secret courrant. Le bloc est validé si la vérification du contrôle est positive.

(1) TYPE est pris ici dans le sens de la partie de l'identifiant d'un bloc ; il détermine donc l'appartenance à un certain jeu secret Mère (ou jeu secret de base).

Deux nouveaux ordres sur le MCS sont nécessaires. Il s'agit d'un ordre de sélection et d'un ordre de production des différentes clés chiffrées.

L'ordre de sélection comprend comme paramètre l'ancien et le nouveau TYPE et un indicateur déterminant les secrets concernés. Trois possibilités existent :

- Le jeu secret et les deux clé émetteur sont générées;
- Le jeu secret et la clé émetteur primaire sont générés;
- La clé émetteur secondaire est générée seule.

L'ordre de sélection provoque automatiquement tous les calculs pour la génération des nouveaux secrets.

L'ordre de calcul est utilisé pour communiquer le nombre aléatoire dans le but de pouvoir réaliser le chiffrement de tous les secrets.

L'ordre de lecture est utilisé à plusieurs reprises (deux lectures par secret) pour lire tous les blocs.

Au niveau de la carte, deux nouveaux ordres sont créés. Il s'agit d'un ordre qui permet de connaître le TYPE du sercret et savoir si la clé émetteur secondaire existe. Un second ordre est ajouté pour l'écriture proprement dite des blocs.

5.4.4. La conception des blocs secrets et le jeu d'instructions

La conception des blocs secret n'est pas fortement changée. La figure 5.9. présente une structure possible du bloc. Une structure ne peut être définie qu'à partir du moment où l'on a choisi l'algorithme cryptographique.

Mot 0	0 1 0	Type	Adresse wi	Indice	NU
Mot 1	0 0				
Mot 2	0 0		Jeu secret Mère		
Mot 3	0 0				
Mot 4	0 X		Contenu du mot d'adresse Wi		
Mot 5	0 X		Contenu du mot d'adresse Wi pour l a divesificationdu jeu secret		
Mot 6	0 X		Adresse Wi JS 1	REF1	REF2
Mot 7	0 X		Adresse Wi JS 2	Zone de contrôle	

Figure 5.9. : La structure du nouveau bloc secret.

Pour le chiffrement, il est nécessaire de pouvoir retrouver le jeu secret de la carte fille. Pour ce faire, on a ajouté dans le bloc le mot d'adresse Wi (Mot 5) utilisé lors de diversification du jeu ainsi que l'adresse Wi (Mot 6 et Mot 7).

Nous avons été un peu étonnés de la redondance d'information entre l'INDICE qui déterminait l'objectif du jeu secret et les bits BD, RD et CD. En effet on peut considérer qu'une affectation automatique de ces zones en fonction de l'INDICE est beaucoup plus sûre pour la sécurité. Nous supprimons donc ces trois indicateurs.

Enfin nous ajoutons une zone de contrôle qui est le reste de la division de tous les bits du bloc, sauf lui-même par un polynôme. Ceci est nécessaire pour l'écriture des blocs secrets. Comme nous l'avons dit au paragraphe 5.4.2., il n'y a plus d'ordre de validation (la validation ne pouvant se faire que par le Mcs émetteur) ; le Mcs dispose ainsi d'un système de contrôle du bloc.

En ce qui concerne les ordres, ils restent relativement les mêmes. Toutefois les ordres d'écriture, de lecture et de validation n'existent plus sauf pour les Mcs du serveur principal (les versions de Mcs sont donc différentes). Nous touverons en plus :

- l'ordre de sélection d'un bloc pour l'écrire dans un autre MCS;
- l'ordre de chiffrement d'un bloc pour son écriture dans un MCS;
- l'ordre de sélection de bloc pour l'écriture du secret dans une carte;
- l'ordre de chiffrement des secrets à écrire dans une carte;
- l'ordre de conversion sera disponible.(pour le transfert de clé en messagerie électronique)
- l'ordre de calcul reste le même.

- l'ordre de lecture de résultat reste le même ; dans certains cas il sera utilisé plusieurs fois pour pouvoir disposer de tout le résultat (cas de génération de clé)

5.5. L'OSA

En ce qui concerne l'OSA, nous ajouterons évidement des instructions qui peuvent exploiter les nouvelles possibilités que nous venons de présenter. Nous ne les développerons pas précisément, le principe étant déjà présenté.

Nous pensons aussi qu'il est intéressant de supprimer certaines redondances comme le chiffrement au niveu du calcul du certificat. De même il n'est pas utile de calculer une clé de chiffrement sur données externes (fonctionalité supprimée).

Un dernier point nous tiend particulièrement à coeur. Pourqoi ne pas imposer une authentification de la carte lors de l'initialisation d'une session. En effet ceci impose automatiquement la présence de la carte, par conséquent, il serait beaucoup plus difficile d'utiliser le OSA dans un but de fraude (lutte contre les attaques internes).

5.6. Le masque de la carte

5.6.1. Introduction

Nous avions précédemment critiqué deux points dans le masque 4. Premièrement le jeux d'instructions qui ne nous semblait pas assez bon sur certains points de sécurité. Nous venons d'y introduire quelques nouvelles instructions ; nous les rapellerons. Deuxièmement, nous avions exprimé quelques réserves sur la découpe en zone de la mémoire EPROM. Nous présenterons une autre organisation.

5.6.2. Les différentes zones de la mémoire EPROM

Nous considérerons deux grands types de zone. Certaines zones sont principalement utilisée par le micro-processeur et d'autre destinées à la mémorisation d'informations de l'application.

Dans les zones concernées directement par le micro-processeur, nous comptons la zone qui comptabilise les accès, la zone de fabrication et la zone où sont mémorisés les différents secrets de l'application.

Ces trois zones doivent être facilement accédées par le micro-processeur. Pour ce faire la zone de fabrication est présente en fin de mémoire, la zone de secrets sera en début de mémoire et la zone d'accès suivant directement la zone de secrets.

Nous avions regretté, dans le masque 4, le manque de souplesse de la mémoire. C'est ainsi que nous avons trouvé utile d'imaginer un descripteur pour chaque zone, reprenant (figure 5.10.) le nombre de mots qu'elle contient, ses droits d'accès et la protection par clé en lecture ou en écriture. Cette protection peut être répartie sur trois sortes de clé : clé émetteur primaire, secondaire ou porteur.

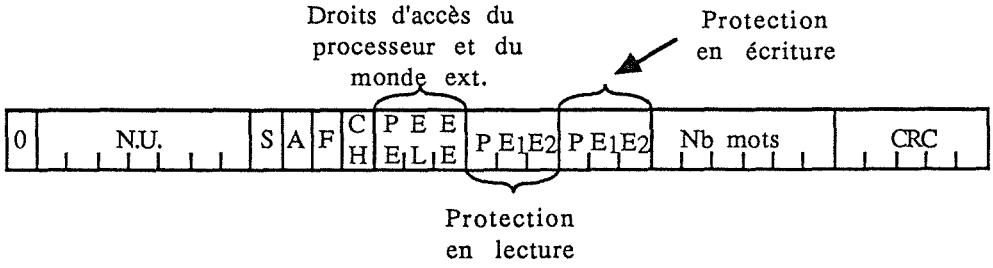


Figure 5.10. : Descripteur de zone.

Toutes les zones sont de longueur totalement variable. Toutefois les trois zones concernant le micro-processeur ont une taille minimum obligatoire.

Pour la zone secrète, la présence de la clé émetteur primaire est obligatoire de même que le jeu secret. Les secrets sont mémorisés dans la zone de fabrication dans une structure de bloc (figure 5.11) où le premier mot comprend le TYPE du secret (numéro donnant l'appartenance à un parc de carte), l'INDICE (permettant de déterminer la "sorte" de la clé : émetteur primaire, secondaire, porteur ou jeu secret) de même qu'une zone de contrôle (reste de la division de tout le bloc, sauf lui-même, par un polynôme). Ce contrôle est essentiel pour l'écriture de nouveaux secrets(1).

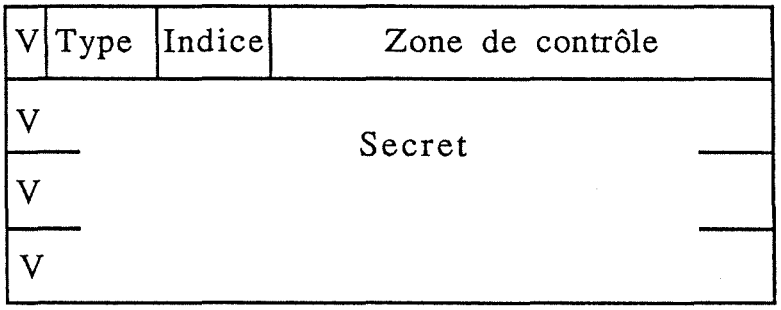


Figure 5.11. : Bloc mémorisant un secret de la carte.

La zone secrète sera remplie à partir des adresses les plus faibles vers les adresses les plus hautes, la prise en compte des secrets se faisant, quant à elle, dans l'autre sens (pour disposer des secrets les plus "jeunes").

La longueur minimale de la zone d'accès ne peut pas être fixée en fonction d'un paramètre précis. En effet la taille dépend de l'utilisation de la carte. Il ne faut pas oublier qu'à chaque présentation de clé un bit, au moins, est utilisé et, lorsque cette zone est totalement utilisée, la carte est arrivée à la fin de sa vie. On peut imaginer une solution à ce problème. Elle consiste, lorsque la zone est complète, à mémoriser dans les deux derniers octets de la zone un pointeur vers une nouvelle zone d'accès qui serait créée. Ceci solutionne le choix difficile de la taille de la zone.

(1) Voir au paragraphe 5.4.3.

La zone de fabrication impose une longueur minimum de deux mots. En effet elle doit mémoriser le numéro de fabricant, le numéro de série de la carte et les indicateurs de la phase de vie de la carte.

5.6.3. Le jeu d'instructions

Nous reprenons ci-dessous les différents ordres que nous avons ajouté, modifié ou supprimé par rapport au Masque 4 :

- l'ordre de présentation du nombre aléatoire sera utilisé. Ce nombre interviendra automatiquement lors de l'utilisation de l'algorithme cryptographique.
- l'ordre de présentation de clé émetteur qui comprend le déchiffrement de la clé par la carte elle même.
- l'ordre de présentation de clé porteur variera selon les options choisies quant au traitement de l'authentification du porteur.
- l'ordre d'interrogation des secrets;
- l'ordre d'écriture de nouveaux secrets;
- l'ordre d'introduction d'un nouveau code porteur (deux possibilités selon la présence d'un mini clavier sur la carte, voir au paragraphe 5.4.3)
- l'ordre de lecture devra éventuellement chiffrer les données selon les paramètres du descripteur de zone, de manière automatique.
- Il n'y a plus de validation de clé en lecture, cette fonction étant réalisée en partie par l'ordre de présentation de clé et l'ordre de lecture.
- l'ordre d'écriture devra aussi, dans certains cas, se charger de déchiffrer les données avant l'écriture proprement dite.
- l'ordre de validation en écriture sera conservé.
- l'ordre de calcul pour les certificats est conservé tel quel.
- Un ordre de chiffrement/déchiffrement sera créé pour solutionner le problème de messagerie électronique.

5.7. Conclusions

Nous venons de présenter plusieurs possibilités. Nous rappelons, comme nous l'avions déjà fait, que la sécurisation dépend totalement de l'application. Dans certains cas, des simples dispositions ou des mesures d'organisation dans l'entreprise peuvent suffire pour solutionner un problème de sécurité. Dans le chapitre suivant, nous allons présenter d'autres perspectives que nous n'avons pas abordées.

Chapitre 6

Autres perspectives

6.1. Introduction

Dans ce chapitre, nous allonc vous suggérer d'autres possibilités que nous n'avons pas abordées dans les chapitres précédents.

6.2. La carte et plusieurs émetteurs

Dans la présentation de la carte masque 4, de même que dans les améliorations que nous avons présentées, nous conservons le principe des deux clés émetteur. Nous avons aussi souligné au chapitre 4 que l'usage de plusieurs cartes peut devenir compliqué, cela à cause de la mémorisation des secrets porteur.

Permettre deux émetteurs pour une carte solutionne déjà quelque peu le problème. Toutefois, on peut remarquer que le masque 4 n'a pas été conçu pour un telle utilisation. En effets le jeu secret de la carte est utilisé indifféremment par les deux émetteurs. Ceci est très gênant pour l'authentification, par exemple, qui se base sur le jeu secret.

Pour ces raisons, de nouveaux masques de cartes sont conçus dans le but de pouvoir faire "cohabiter" des applications différentes. La carte se trouve hiérarchisée à trois niveaux.

L'étude de telles cartes est très intéressante. Plusieurs choses peuvent se heurter à son développement. On peut retenir principalement la taille mémoire ainsi que le problème commercial : une banque accepterait-elle d'utiliser des cartes qui ne portent pas son logo ?

On peut noter que les principes de sécurité présentés au chapitre 5 restent d'application pour ce type de carte. La différence principale réside dans la hiérarchisation de la mémoire.

6.3. Le terminal à plusieurs applications.

On peut considérer qu'un terminal peut, lui aussi, fournir des services réalisés par des émetteurs différents. Ceci s'approche de l'utilisation de la carte pour des services accessibles à partir d'un Minitel. Un développement de techniques pareilles est certainement profitable à l'augmentation des services que la carte peut offrir.

6.4. L'utilisation "off-line" de la carte

Jusqu'à présent, nous nous sommes basés dans ce mémoire sur un réseau où le terminal est toujours connecté au serveur, au moins pendant le traitement de la carte.

La carte permet en effet d'être utilisée sans que le terminal soit connecté au serveur. Le mécanisme se base sur le fait qu'il est possible de mémoriser une information dans la carte et que, une fois écrite, elle ne peut plus être modifiée.

Par exemple on peut imaginer, dans le cas bancaire, qu'un distributeur de billets (faisant fonction de terminal) n'est pas connecté au serveur. Le distributeur délivre l'argent si la demande ne dépasse pas un montant inscrit dans la carte. Ce dernier est diminué de la somme retirée. Lorsqu'il a atteint une somme inférieure au besoin du porteur, ce dernier utilisera un terminal qui peut se connecter au serveur qui déterminera un nouveau montant disponible. Pour une telle réalisation il est nécessaire que le distributeur dispose d'un MCS où au moins un bloc pour l'émission de la clé émetteur ainsi qu'un bloc de contrôle de certificat existent.

Il est intéressant d'étudier les différentes implications de l'insertion des MCS dans les terminaux. De même il est essentiel d'effectuer des recherches au niveau du développement des applications dans le but de minimiser l'utilisation de la mémoire de la carte et par conséquent maximiser, sa durée de vie. Des techniques

telles que la signature électronique d'une transaction peuvent être très intéressantes dans la mesure où elle n'utilise pas de mémoire.

6.5. Prouver la bonne construction d'une carte ?

En matière de sécurité, un bon principe est de mettre tout en doute. Par conséquent suspectons le constructeur de la carte[1]. On peut aussi imaginer des procédures qui permettent à ce constructueur de prouver son honnêteté. Pourqoi, par exemple ne pas publier sous copyrigth toutes les sources des masques des cartes, PSA, OSA, etc. Pour vérifier la bonne implémentation sur la carte, un ordre permettrait de lire toute la ROM. Le développement de tels mécanismes peut se révéler indispensable pour la sécurisation d'applications à très haute sécurité.

6.6. Conclusion

Nous arrêtons ici ces différentes suggestions. Leur but est de montrer que la carte à micro-processeur a un très bel avenir devant elle. Beaucoup de perspectives se présentent à tous les niveaux : hardware de la carte, software dans la conception de son masque et dans la conception de tous les logiciels encadrant l'utilisation de la carte.

(1) Dans le terme "constructeur" nous reprenons aussi bien l'entreprise en temps que personne morale mais aussi les employés et ouvriers qui profiteraient de leur position pour produire des cartes "truquées"

Conclusion

Nous avons présenté dans ce mémoire la carte à micro-processeur dans ses principes de base, utilisée dans le cadre d'une application avec serveur. Cette présentation était calquée sur la carte masque 4 qui est d'un conception plutôt ancienne. Nous l'avons critiquée assez bien quant à la protection de ses secrets.

Nous avons regretté durant la réalisation de ce mémoire, le manque flagrant de publications dans les domaines qui concernent la carte à micro-processeur. Nous pensons, par exemple, à des protocoles de distribution de clés de chiffrement pour algorithme à clé secrète, technique de diversification, etc. Nous estimons que cette absence peut nuire fortement à la qualité des différentes conceptions, toute publication permettant un échange d'idées qui ne peut être que positif pour le développement de la carte. Ceci revient quelque peu à la suggestion faite au paragraphe 6.5 qui est de publier toutes les productions de logiciels. Cette pratique permet un "contrôle" et une "participation" des utilisateurs à l'amélioration de la carte.

Dans ce mémoire nous nous sommes attachés à améliorer le masque de la carte seulement dans le sens de la sécurité. Nous nous sommes moins occupés d'améliorations portant sur la souplesse de la carte telles que la hiérarchisation de la mémoire pour pouvoir accepter différents services. Nous avons plutôt développé des mécanismes qui garantissent mieux la sécurité des secrets d'une application (chiffrement des clés, système de communication de clés de chiffrement pour le courrier électronique).

Cette étude est loin d'être complète. Elle permet toutefois de se faire une idée sur le niveau de sécurisation qu'offre la carte à micro-processeur. Une étude plus poussée par exemple au niveau cryptographique serait très intéressante.

Dans le dernier chapitre, nous avons montré que la carte a encore "de beaux jours" devant elle. Il ne faut toutefois pas oublier qu'en principe de sécurisation, c'est toujours une "course" entre les concepteurs de solutions de sécurisation et les attaquants. Il est donc important de ne négliger aucun point, que ce soit du développement hardware ou software. N'oublions pas que "la résistance d'une chaîne est égale à la résistance de son plus faible maillon". Ce principe est tout à fait applicable aux problèmes de sécurité.

Bibliographie

[AKL 83] Selim G.Akl, "Digital Signatures : A tutorial survey", Computer, février 1983, 15-24.

[DUR 88] Jean-Jacques Durré, "La sécurité informatique, une question de survie économique", Pourquoi Pas ? 11/08/88, 19-20.

[M4 87] "Guide utilisateur des carte CP8 Masque 4 et 8", BULL CP8, Réf.BULL CP8 : TU 0047 F02, Juillet 1987.

[MCS 87] "Guide utilisateur du MCS, module de contrôle et de sécurité", BULL CP8, Réf.BULL CP8 : TU 0066 F02, Juin 1987.

[PSA 87] "PSA, Processeur de Sécurité Associé à un serveur, spécifications d'interface du PSA 8 cartes", BULL CP8, Réf.BULL CP8 : TU 0033 F03, Juillet 1987.

Nous avons aussi consulté différents documents de la société Bull CP8. Ils reprennaient les sujets suivants :

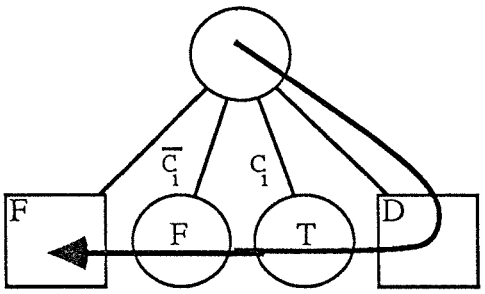
- spécification cartes Masque 4
- spécification cartes Masque B0
- spécification cartes Masque B2
- spécification du Mcs
- spécification de la carte PDS 44

Annexes

Présentation de l'arbre programmatique

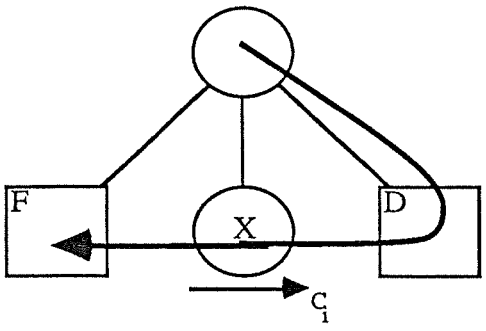
Cet outil de dévelloppement permet de décrire la structure d'un logiciel en faisant ressortir uniquement les tests et les boucles.

La représentation du test est la suivante :



C_i se reporte à une liste de conditions. Les boites D et F représentent respectivement les instructions à effectuer avent et après le test. Notez bien le sens de lecture qui est de haut en bas et de droite à gauche ! Les instructions symbolisées par le cercle marqué d'un V seront exécutée si la condition C_i est réalisée car on note la présence de C_i sur l'arc qui porte sur ce cercle. $\overline{C_i}$ est la négation de C_i. et porte par conséquent sur le cercle marqué d'un F qui représente les instructions au cas où la condition n'est pas respectée.

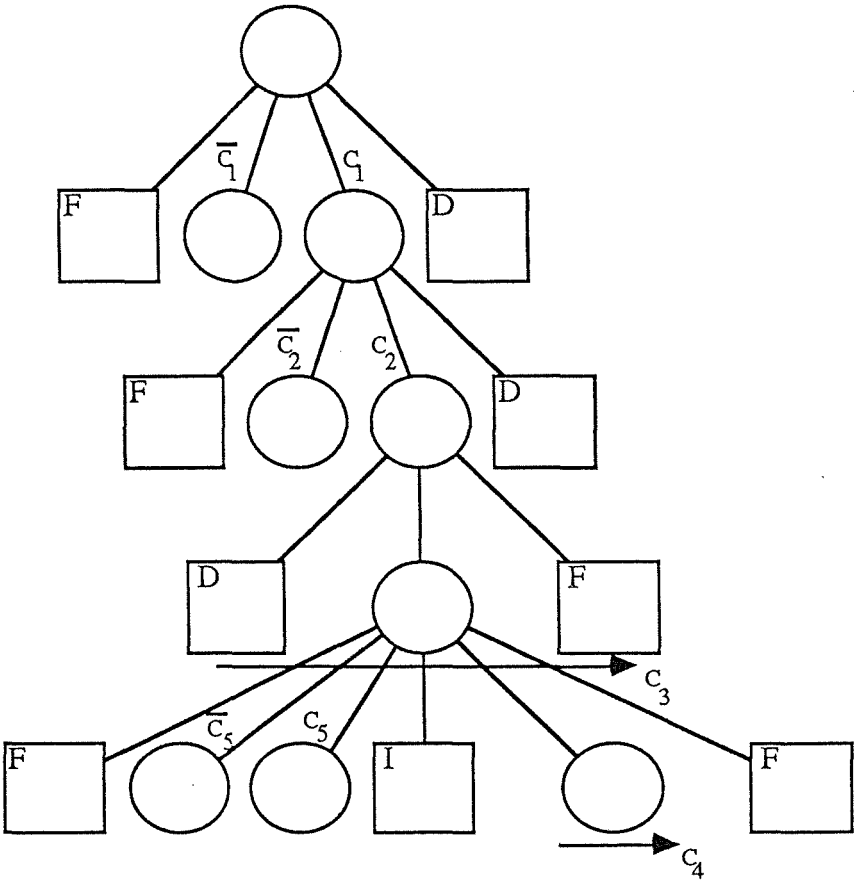
La représentation de la boucle est la suivante :



Le même principe que le test est utilisé. C_i est la condition d'une boucle de type DO WHILE. Le rectangle D représente les instructions qui sont à exécuter

avent la boucle, par exemple l'initialisation de conteurs, d'indicateurs, etc. Dès que la condition C_i ne sera plus remplie à la fin des instructions représentées par le cercle X, les instructions du rectangle F sont exécutées.

C'est les deux structure de base de cet outil. Il suffit de les combiner pour représenter un programme complet. Ci-dessous nous vous présentons une structure possible d'un logiciel.



On peut remarquer qu'en composant les deux structures de manière séquentielle, un rectangle I (I pour Interface) apparaît. Il est en fait la concaténation du rectangle de fin de la première structure et du bloc de début de la seconde structure.

liste des codes retour du OSA

Code en hexadécimal	Description
40	L'opération s'est bien effectuée.
42	Il n'y a pas de MCS pour réaliser la fonction demandée.
43	Message incorrect, erreur de longueur ou code opération inconnu.
45	Numéro de session déjà existant.
46	Numéro de session n'existant pas.
47	Erreur de type de carte.
4A	Erreur de séquencement.
4B	Mot d'état différent de 90 00.
51	Erreur accès MCS numéro 1
52	Erreur accès MCS numéro 2
53	Erreur accès MCS numéro 3
54	Erreur accès MCS numéro 4
55	Erreur accès MCS numéro 5
56	Erreur accès MCS numéro 6
57	Erreur accès MCS numéro 7
58	Erreur accès MCS numéro 8